VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2017 Petr Šmejkal

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

APLIKACE PRO NOSITELNOU ELEKTRONIKU SE SYSTÉMEM ANDROID WEAR
APPLICATIONS FOR WEARABLE DEVICES BASED ON ANDROID WEAR OS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Petr Šmejkal**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Jiří Hošek, Ph.D.**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

Student: Petr Šmejkal ***ID:*** 173758
Ročník: 3 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Aplikace pro nositelnou elektroniku se systémem Android Wear

POKYNY PRO VYPRACOVÁNÍ:

Specifickou oblast IoT představují takzvané nositelnosti (wearables), které jsou dnes velmi často reprezentovány chytrými náramky nebo hodinkami. V rámci této bakalářské práce bude pozornost soustředěna na platformou Android Wear a vývoj aplikace pro ovládání klíčových funkcí spárovaného mobilního telefonu (zjišťování informací o parametrech WiFi/3G/4G připojení, zobrazení aktuální polohy s využitím GPS, notifikace o stavu baterie, atd.).

DOPORUČENÁ LITERATURA:

[1] RUIZ, David Cuartielles. 2014. Professional android wearables. Indianapolis, IN: John Wiley and Sons. ISBN 11-189-8685-7.

[2] YE, Roger. 2016. Embedded programming with Android: bringing up an Android system from scratch. New York: Addison Wesley. ISBN 01-340-3000-1.

Termín zadání: 1.2.2017 ***Termín odevzdání:*** 8.6.2017

Vedoucí práce: doc. Ing. Jiří Hošek, Ph.D.
Konzultant:

doc. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Bakalářská práce na téma „Aplikace pro nositelnou elektroniku se systémem Android Wear“ popisuje základní principy komunikačních sítí – M2M (Machine-to-Machine), H2H (Human-to-Human) a D2D (Device-to-Device). Práce se dále zabývá nositelnými zařízeními, zejména chytrými hodinkami a dostupnými operačními systémy, kdy je pozornost soustředěna na systém Android/Android Wear. V praktické části je popsaná funkcionalita, vzhled a struktura vyvinuté aplikace pro Android Wear.

KLÍČOVÁ SLOVA

Android aplikace, Android Wear, D2D, chytré hodinky, M2M, nositelné zařízení

ABSTRACT

The bachelor thesis “Application for Android Wear wearables” describes key principles of the communication networks – M2M (Machine-to-Machine), H2H (Human-to-Human) and D2D (Device-to-Device). It is also focused on wearable devices especially smartwatches, available operation systems for smartwatches where the attention is paid to Android/Android Wear operating systems. Practical part of bachelor thesis details design and structure of developed application for target Android Wear smartwatch.

KEYWORDS

Android application, Android Wear, D2D, M2M, smartwatch, wearables

ŠMEJKAL, Petr. *Aplikace pro nositelnou elektroniku se systémem Android Wear*. Brno, 2017, 66 s. Semestrální projekt (BB2T). Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: doc. Ing. Jiří Hošek, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svůj semestrální projekt na téma „Aplikace pro nositelnou elektroniku se systémem Android Wear“ jsem vypracoval(a) samostatně pod vedením vedoucího semestrálního projektu a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedeného semestrálního projektu dále prohlašuji, že v souvislosti s vytvořením tohoto semestrálního projektu jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu semestrální práce panu doc. Ing. Jiřímu Hoškovi, Ph.D. a panu Ing. Pavlu Maškovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno . . . .

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v tomto semestrálním projektu byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno ....

podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM VÝPISŮ

ÚVOD

Nositelná zařízení existují již více než deset let v podobě chytrých hodinek, sluchátek, zařízení sledujících pohybové aktivity nebo vitální funkce a kamery. Ovšem až v posledních pár letech dochází k jejich rozmachu mezi koncovými zákazníky.

Společnost Google představila možnosti a funkcionality technologie chytrých brýlí v roce 2012 a i nadále v tomto oboru pokračovala uvolněním systému Android Wear v roce 2014. Android Wear je Android operační systém pro nositelné zařízení, který nabízí API (Application programming interface) pro výměnu dat s ostatními zařízeními a umožňuje použití Google služeb jako jsou mapy, nebo Google disk [1].

Tato bakalářská práce je ve své teoretické části především zaměřena na chytré hodinky a operační systém Android Wear. V práci je rozebrán životní cyklus Android aplikací, možnosti Android API (získání informací o Wi-Fi připojení, ovládání Wi-Fi, Bluetooth, GPS, ...). Dále je rozebrána Data Layer API tj. část Google Play služeb, která nabízí funkce pro výměnu dat mezi zařízeními, požadavky na vývoj pro Android Wear a ovládání vývojového prostředí Android Studio.

Praktická část se zaobírá vytvořenými aplikacemi pro chytré hodinky s operačním systémem Android Wear. Byly vytvořeny dvě aplikace. První nazvaná Toggles, která umožňuje ovládání základních funkcí telefonu (Wi-Fi, Bluetooth, zvuk, ...), automatickou synchronizaci jejich stavu, zobrazení informací o aktuálním Wi-Fi připojení, stavu baterie a pozice na mapě včetně vypsání souřadnic polohy. Druhá aplikace nazvaná Bluetooth manager, která funguje pouze na straně chytrých hodinek a získává informace z Bluetooth adaptéru. Mezi tyto informace patří seznam připojených, spárovaných a nalezených zařízení.

1 KOMUNIKAČNÍ PRINCIPY V BEZDRÁTOVÝCH SÍTÍCH

V této kapitole jsou popsány koncepty typů komunikace H2H (Human-to-Human), M2M (Machine-to-Machine), D2D (Device-to-Device) a nositelná zařízení (wearables), která reprezentují komunikaci typu D2D. Bude také popsán vývoj trhu M2M zařízení a nositelných zařízení.

1.1 H2H

Za komunikaci H2H neboli člověka s člověkem lze považovat každou komunikaci, která je uskutečněna mezi dvěma osobami prostřednictvím zařízení (telefon, počítač, ...). Při telefonování musí být zajištěna kvalita služby neboli QoS (Quality of Service) – nízké přenosové zpoždění, malé kolísání zpoždění, nízká ztrátovost [2]. V současnosti je velmi často diskutována také kvalita zážitků QoE (Quality of Experience), která se zaměřuje na zákazníkovu zkušenost se službou. QoE tedy reprezentuje QoS z pohledu zákazníka [3].

1.2 M2M

Technologie M2M umožňuje zařízením jednosměrnou či obousměrnou komunikaci bez interakce člověka. Zařízení mohou v síti komunikovat po jakémkoliv komunikačním médiu například sériový kabel, bezdrátový přenos nebo Ethernet. Jedná se převážně o senzorická zařízení jako teploměr, elektroměr, meteorologické stanice a další. Tyto zařízení jsou často zapojeny v Internetu a tím utvářejí koncept zvaný IoT (Internet of Things) nebo IoE (Internet of Everything), který má v dnešní době velký potenciál. Do Internetu věcí mohou být zařízení připojena přímo nebo přes ústřední prvek, který se chová jako agregační brána [4].

1.3 D2D

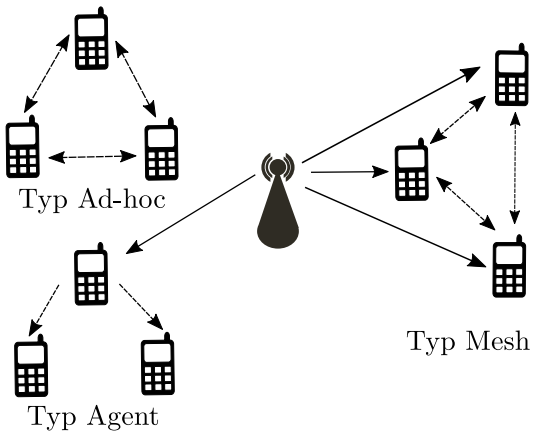
Technologie D2D umožňuje přímou komunikaci dvou či více zařízení bez využití telekomunikační sítě za předpokladu, že jsou v dosahu [5]. Toto paradigma je využíváno u bezdrátových technologií už několik let. Mnoho lidí si pomocí Bluetooth páruje své telefony s chytrými hodinkami či fitness náramky, které jsou dnes všeobecně nazývané wearables neboli nositelná zařízení. Také je využívána technologie Wi-Fi Direct pro výměnu většího množství dat dvou zařízení. Avšak v současnosti je předním tématem D2D nasazení v mobilních sítích a vytvoření tak hybridní mobilní sítě. V LTE

(Long Term Evolution) od specifikace 3GPP release 12 je implementovaná možnost sestavit přímé datové spojení, kdy však signalizace stále probíhá skrze mobilní síť [6]. Výhodami tohoto přístupu mohou být, snížení zátěže sítě, rychlejší komunikace zařízení v dosahu nebo v případě výpadku základnové stanice přemostění signálu přes jiné zařízení k dostupné stanici. Na druhou stranu je potřeba vytvořit protokoly zajišťující signalizaci, nalezení zařízení v dosahu nebo směrování dat. V LTE se touto problematikou zabývá ProSec (Proximity Service) [5].

Typy D2D

Na základě přístupu ke spravování D2D komunikace po stránce řízení přístupu a objevování zařízení můžeme D2D technologii v LTE rozdělit na několik typů viz obr. 1.1 [5].

1. Ad-hoc: V tomto případě jsou zařízení nacházející se v dosahu schopna autonomně vytvořit, takzvaný komunikační cluster (shluk, skupinu), ve kterém si mohou vyměňovat data. Vše se děje nezávisle na základnové stanici telekomunikační sítě BTS (Base Transceiver Station) nebo eNodeB tj. evolved NodeB.
2. Agent: Pokud požaduje několik zařízení z jednoho clusteru stahovat stejný obsah (video, hudbu, ...), je jedno ze zařízení zvoleno agentem, přístupovým bodem clusteru. Tento agent stáhne data z Internetu a dále je rozdistribuuje ostatním zařízením.
3. Mesh (mříž): Vytvoření clusteru probíhá stejně jako u předešlých dvou typů. Avšak požadovaná data více zařízeními je rozdělena na několik fragmentů, kdy každý z nich je stažen jedním zařízením. Fragmenty jsou následně přeposlány v rámci clusteru.



Obr. 1.1: Přehled typů D2D komunikace (Mesh, Agent, Ad-hoc) [5].

1.4 Srovnání komunikačních principů

Požadavky zařízení H2H, M2M a D2D se liší v několika parametrech například rychlosti přenosu dat, minimálního zpoždění, množství přenášených dat. Srovnání těchto parametrů je uvedeno v tab. 1.1.

1.5 Historický vývoj

1.5.1 H2H

Již od doby, kdy začalo docházet k civilizování lidské rasy, k zakládání vesnic či měst a velkých říší, byla klíčovou potřebou rychlá komunikace na větší vzdálenosti. Tato potřeba byla hybatelem k vývoji potřebných technologií. Za první významné technologické kroky můžeme považovat vytvoření prvního elektronického telegrafu roku 1836 (Samuel Morse, Joseph Henry, Alfred Vail), otevření první telegrafní linky roku 1844 mezi městy Washington a Baltimore a úspěšné položení transatlantické telegrafní linky roku 1866. Druhým velice významným objevem byla možnost přenášet hlas díky Alexanderu Grahamovi Bellovi, který roku 1876 vynalezl telefon. Neustálý tlak na zefektivnění, zrychlení a zlevnění telekomunikační služeb vedl k takovému technologickému pokroku, který umožnil automatickou komunikaci zařízení. Začalo tak docházet k přizpůsobování telekomunikačních linek pro tyto služby [7].

1.5.2 M2M

První technologie naplňující koncept M2M se objevila již na začátku druhé poloviny dvacátého století. Roku 1968 byl nalezen způsob, který umožnil při navázání hovoru přenést dodatečné informace a to identifikátor volajícího. S příchodem GSM (Global System for Mobile communications) standardu a jeho plošnou aplikací došlo k rozvoji různých typů senzorických zařízení, které k odesílání dat a alarmů využívají právě GSM síť. Roku 1999 došlo k velkému rozšíření technologie RFID (identifikace pomocí rádiových frekvencí), s kterou se dnes běžně setkáváme v nákupních centrech. K největšímu rozvoji M2M došlo díky konceptu IoT tj. napojení zařízení do Internetu. Přestože koncept IoT je znám od roku 1999, začal být populární až roku 2010 a zařízení naplňující tento koncept se dostala na trh až roku 2014. V současnosti je k dispozici několik řešení chytrých domácností, ovládání vytápění, zabezpečovacích systému a mnoho dalších [8].

Tab. 1.1: Srovnání H2H a M2M [9].

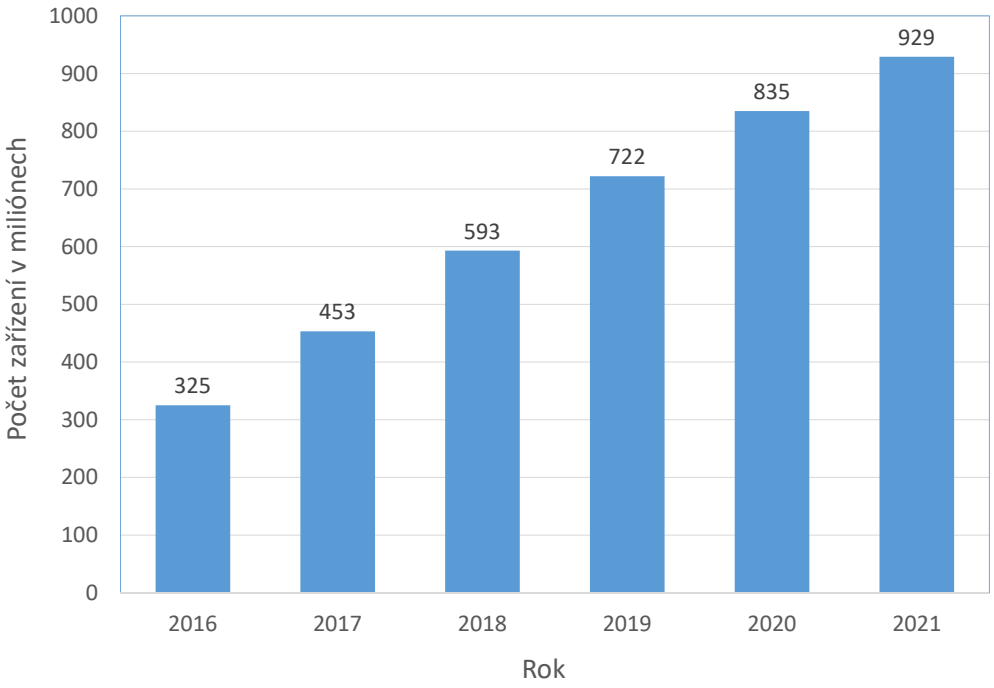
Parametry	H2H	M2M
Počet zařízení	Koncem roku 2015 bylo 7,9 miliard mobilních zařízení a je odhadováno, že toto číslo vzroste na 11,6 miliardy [10].	Počet zařízení cca 0,9 miliard a do roku 2020 je očekávaný nárůst na 3,1 miliardy [10]. Tento fakt je jedním z důvodů pro zavedení IPv6.
Množství dat	Většina přenesených dat je směrem ke koncovému uživateli v podobě souborů, generování HTTP(s) provozu nebo hlasového či obrazového streamu.	Datový přenos převážně od koncového zařízení požadující malou šířku pásma. Kamerové systémy mohou vyžadovat větší šířku pásma v závislosti na kvalitě multimediálního obsahu. Video zakódované v H.264 s rozlišením 720p a 30 fps (frames pre second) má datový tok 1,2 - Mb/s.
Baterie (energetická náročnost)	Baterii lze kdykoliv nabít a u většiny zařízení i vyměnit.	Zařízení by mělo být schopné samo generovat energii nebo alespoň na baterii vydržet několik měsíců či let v závislosti na typu zařízení. Zařízení jako vodoměry, elektroměry, plynoměry vyžadují výdrž baterie až desítky let v případě využití komunikačního protokolu Wireless M-BUS.
Zpoždění	Nízké zpoždění je tolerováno (u hlasu maximálně 150 ms).	Zpoždění u aplikací pracujících v reálném čase by nemělo přesáhnout 250 ms. U zařízení jako jsou například meterologické stanice nebo teploměry v chytré domácnosti může být zpoždění až v rámci několika sekund.
Zabezpečení	Odcizení nebo závada zařízení je rychle odhalena, protože člověk tyto zařízení nosí u sebe nebo s nimi přijde do kontaktu několikrát denně. Díky tomu je možné rychle zablokovat nebo opravit zařízení.	Zařízení by měla být zabezpečena proti odcizení nebo neautorizovanému přístupu. Z důvodu umístění ve špatně dostupných či hůře kontrolovatelných prostorách.

1.5.3 D2D

První komerční a rozšířenou technologií naplňující koncept D2D je IrDA (přenos signálu v infračerveném spektru), která byla poprvé standardizovaná roku 1994. Rychlost přenosu technologie IrDA zprvu dosahovala 115 Kb/s, ale v současnosti dosahuje rychlosti až 4 Mb/s a komunikace může probíhat maximálně do vzdálenosti 1 m bez překážek [11]. Další významnou technologií je Bluetooth, která byla poprvé standardizovaná roku 1999. Bluetooth vysílá signál všesměrově v rádiovém spektru na frekvenci 2,4 GHz. Od verze v3.0 podporuje rychlost přenosu až 24 Mb/s do vzdálenosti 10 m. Od verze v4.0 podporuje nízko energetiký přenos a je nazýván BLE (Bluetooth Low Energy). Bluetooth má oproti IrDA několik výhod. Vysílá všesměrový signál, který může procházet překážkami, přenos dat je rychlejší a má větší dosah. S neustále rostoucím požadavkem na rychlost přenosu, z důvodu zvyšujícího se množství dat, vznikly nové technologie a jedou z nich je i Wi-Fi Direct. Wi-Fi Direct dosahuje rychlosti 250 Mb/s až do vzdálenosti 10 m. V současnosti je Bluetooth hodně využíváno a to zejména s nositelnou elektronikou jako jsou chytré hodinky, fitness náramky a další [12].

1.6 Vývoj trhu

V současnosti dochází k růstu popularity IoT/IoE, který potvrzuje významný nárůst chytrých zařízení a M2M spojení. Zařízení M2M zajišťující zabezpečení a automatizaci budov, monitorování a navigaci aut nebo sledování fyzického stavu osob a další jsou využívány jak v průmyslu tak i v spotřebitelském segmentu. Firma Cisco ve svém výhledu uvádí, že počet všech M2M spojení od roku 2015 vzroste do roku 2021 o 2,7 miliardy na 3,3 miliardy [10]. Důležitým faktorem přispívající k nárůstu popularity IoT/IoE je vznik nositelných zařízení, které mají velký potenciál ve zdravotnictví, službách založených na lokalizaci a virtuální realitě. Cisco odhaduje, že v roce 2021 bude po světě 929 miliónu nositelných zařízení, což je téměř trojnásobný nárůst od roku 2016 viz obr. 1.2. Očekává se, že pouze 7,4 % všech nositelných zařízení bude disponovat mobilním připojením k Internetu. Nárůst počtu nositelných zařízení podle regionů lze vidět v tab. 1.2. Důsledkem enormního nárůstu těchto zařízení dojde k signifikantnímu znásobení datového toku. Průměrná velikost přenesených dat M2M a nositelných zařízení za měsíc je sedmkrát větší než u mobilních telefonů. V roce 2015 bylo průměrně přeneseno 164 MB/měsíc u M2M zařízení a 153 MB/měsíc u nositelných zařízení. Do roku 2020 je očekáván nárust na 670 MB/měsíc a 558 MB/měsíc [10].



Obr. 1.2: Graf počtu připojených nositelných zařízení [10].

Tab. 1.2: Nárust nositelných zařízení dle oblastí [10].

	2016		2021	
Oblast	Počet zařízení v miliónech	Globální zastoupení	Počet zařízení v miliónech	Globální zastoupení
Asie a Tichomoří	99,8	31 %	258,2	28 %
Centrální a východní Evropa	17,5	5 %	55,6	6 %
Latinská Amerika	12,6	4 %	39,0	4 %
Střední východ a Afrika	14,0	4 %	37,5	4 %
Severní Amerika	127,1	39 %	378,8	41 %
Západní Evropa	54,3	17 %	159,7	17 %
Celkem	325,3	100 %	928,3	100 %

2 NOSITELNÁ ZAŘÍZENÍ

Wearables neboli nositelná zařízení jsou velkým předmětem diskuze v současnosti. Nositelná zařízení jsou malé elektronická zařízení vybavená mikročipem, které mohou obsahovat senzory schopné měřit stav našich vitálních funkcí či pohyb. Jak již z názvů vyplývá, jedná se o nošená zařízení, a proto by měla být pohodlná a lehce připevnitelná na tělo nebo oblečení. Takovými zařízeními jsou například chytré hodinky, náramky či brýle [13]. Technologický vývoj pokročil natolik, že je možné pomocí hodinek zjistit stav telefonu, přečíst si zprávy, zavolat a přijmout notifikace z mobilních aplikací aniž by bylo nutné vytáhnout mobilní telefon z kapsy. S některými modely je možné zobrazovat webové stránky. Toho všeho je možné dosáhnout pomocí Bluetooth nebo Wi-Fi, pokud jsou zařízení spárována [1].

2.1 Druhy nositelných zařízení

Mezi nejvýznamnější a nejrozšířenější typy nositelných zařízení patří chytré náramky a hodinky, ale vývoj probíhá i v jiných oblastech jako jsou chytré brýle, kontaktní čočky nebo nebo chytré boty. U zařízení je zapotřebí spárování s jiným zařízením jako je mobilní telefon nebo tablet za účelem synchronizace dat a jejich zobrazení, základní nastavení nebo upozornění na definovanou událost [1].

2.1.1 Fitness náramky

Mezi hlavní funkce fitness náramků patří sledování pohybové aktivity. Díky zabudovaným senzorům jsou schopny relativně přesně spočítat množství kroků, rozhodnout zda se jedná o chůzi nebo běh. Některé náramky obsahují i senzor srdečního tepu a mohou tak monitorovat v jaké fázi spánku se nacházíme a vzbudit nás ve vhodnou dobu pro naše tělo [14]. Některé modely, například Mi Band 2 od firmy Xiaomi[1] obsahují i možnost upozornění hovoru, příchozí zprávy a dalších událostí, pokud jsou spárované například s mobilním telefonem. Pro komunikaci je nejčastěji používána technologie BLE. Výdrž baterie velice závisí na typu technologie displeje,jeho velikosti a doby komunikace se spárovaným zařízením. Výdrž baterie může dosahovat až čtyř týdnů. Tato zařízení mají díky senzoru měření tepu velký potenciál ve zdravotnických oblastech, protože mohou poslat upozornění při srdečních obtížích [1].

[1] `http://www.mi.com/en/miband2/`

2.1.2 Chytré hodinky

Možnosti chytrých hodinek závisí zejména na využité platformě (operačním systému), ale lze říci, že obsahují přinejmenším stejné funkce jako chytré náramky. Některé hodinky, zejména ty přizpůsobené pro sportovce, mohou obsahovat také GPS modul, barometr nebo teploměr. Nejvíce možností poskytují hodinky s dotykovým displejem, na kterých běží například platformy Android Wear nebo Watch OS (Apple). Tyto platformy nabízí mnoho možností, díky dobrému SDK (Software Development Kit) a API (Application Programming Interface) rozhraní pro komunikaci s mobilním telefonem. Zaslaná upozornění mohou obsahovat obrázky i text a tím co nejblíže přiblížit o co se jedná, dokonce je možné na ně zareagovat přímo z hodinek a tak například ukončit hovor nebo odpovědět na textovou zprávu. Pomocí aplikací je možné ovládat přímo funkce chytrých zařízení (zapnou vypnout Wi-Fi, zvuk a mnoho dalších). Je možné vytvořit aplikace, které běží nezávisle na spárovaném zařízení. V případě, že mobilní telefon obsahuje Wi-Fi modul, obdrží notifikace i když není zařízení v dosahu spárovaného zařízení. Některé modely obsahují NFC (Near Field Communication) modul, takže lze využít chytré hodinky i jako platební kartu [1].

2.1.3 Chytré brýle

V současnosti jsou chytré brýle spíše rozšířením chytrých telefonů, které běžně umožňují zobrazit upozornění, hlasem vyvolat akce (spuštění aplikace, odeslání zprávy, atd.), nahrávat video či zvuk. Nejzajímavějším využitím chytrých brýlí je rozšířená (Augmented Reality) a virtuální (Virtual Reality) realita. Rozšířená realita je označení pro obohacení reálného obrazu o virtuální objekty nebo transformace obrazu například pomocí změny jasu či barev. Rozšířené reality lze například využít při modelování 3D objektů nebo navrhování interiérů. Virtuální realita vytváří virtuální obraz reálného nebo nereálného místa, se kterým lze interagovat [1].

2.1.4 Ostatní nositelná zařízení

Společnost Google pracuje na vývoji kontaktních čoček, které by umožnily sledovat hladinu cukru v krvi a v případě přesáhnutí nebo klesnutí nastavené hladiny varovaly pacienta například pomocí zabudované led kontrolky [15]. Vývoj probíhá i oblasti chytrého oblečení za účelem monitorování zdravotního stavu, zejména monitorování srdečního tepu nebo jako módního doplňku pomocí optických vláken a LED diod [1]. Jednoho dne by lidé mohli nosit chytré boty se zabudovanou GPS, které by je pomocí vibrací naváděly k cíli [15].

2.2 Druhy platforem chytrých hodinek

Mezi nejrozšířenější platformy chytrých hodinek patří Watch OS, Android Wear, Pebble a Tizen. Přehled jejich podporovaných funkcí je v tab. 2.1.

2.2.1 WatchOS

WatchOS je platforma od firmy Apple, která nabízí mnoho funkcí. Bohužel třetím stranám a vývojářům nenabízí všechny tyto funkce. Ve verzi WatchOS 2 je zpřístupněno třetím stranám používání mikrofónu, akceleometru a dalších, ale API pro ovládání Bluetooth nebo NFC zůstaly nedostupné. Aplikace běží kompletně na hodinkách, které by měly být spárované s iPhonem, jinak nabízí velice omezené funkce. V září 2016 byla uvolněna nová verze WatchOS 3, která přinesla změny v oblasti vzhledu a zlepšení výkonosti [16]. Vyvíjet pro tuto platformu je možné v programovacím jazyku Swift nebo Objective-C [17], [18].

2.2.2 Android Wear

Android Wear je bezplatný operační systém pro chytré hodinky, který je používán různými výrobci LG, Asus nebo Sony. Mezi zařízení hostící tuto platformu patří LG G Watch[2], Asus Zenwatch 2[3] a Sony Smartwatch 3[4]. Začátkem února 2017 byla uvolněna nová verze Android Wear 2.0, která poskytuje mnohem více svobody vývojářům. Mezi přední novinky patří možnost běhu nezávislých aplikací, které mohou fungovat bez spárovaného telefonu nebo tabletu. Další významnou funkcí je možnost aplikací komunikovat s telefonem pomocí Wi-Fi z rozdílných sítích za předpokladu, že hodinky obsahují Wi-Fi modul. Vývoj aplikací je možný s Android Wear SDK a programovacího jazyku Java [17], [18].

2.2.3 Pebble OS

Všechny hodinky, na kterých beží tento systém, obsahují bezdotykové displeje s elektronickým inkoustem a Bluetooth. Mohou být provázány s iPhone i Android chytrými telefony, ale nenabízí tolik funkcí jako WatchOS a Android Wear. Na rozdíl od WatchOS a Android Wear má možnost použití takzvaných „smartstraps“, elektroniky umístěné v pásku, která umožňuje komunikovat s hodinkami a rozšířit jejich možnosti. Vývíjet pro Pebble OS je možné pomocí programovacích jazyků C nebo JavaScript [17], [18].

[2] http://www.lg.com/us/smart-watches/lg-W100-lg-watch
[3] https://www.asus.com/ZenWatch/ASUS-ZenWatch-2-WI501Q
[4] http://www.sonymobile.com/cz/products/smart-products/smartwatch-3-swr50

2.2.4 Tizen

Tizen je otevřený operační systém postavený na linuxovém jádře, který pohání různé typy zařízení: počítače ve vozidlech, chytré televize nebo nositelná zařízení. Firma Samsung zvolila právě tuto platformu pro některé chytré hodinky. Tato platforma sice není rozšířená, ale nabízí vývojářům přístup ke kompletní Tizen API, včetně NFC a Bluetooth. Je možné spárování jak s iPhone tak i Android chytrým telefonem. Podporuje také běh nezávislých aplikací. Vyvíjet aplikace je možné pomocí HTML5, CSS, JavaScript, C a C++ [17], [18], [19].

2.2.5 Ostatní platformy

Existují i jiné operační systémy pro chytré hodinky, ale nejsou tak rozšířené jako výše uvedené. Příkladem jsou AsteroidOS [20], WebOS [21] nebo RTOS (Real time operating system) [22].

Tab. 2.1: Srovnání platforem pro chytré hodinky [23], [24], [25], [26].

funkce	WatchOS 3	Android Wear	Pebble	Tizen
Změna vzhledu	Ano	Ano	Ano	Ano
Notifikace	Ano	Ano	Ano	Ano
Aplikace třetích stran	Ano	Ano	Ano	Ano
Synchronizace přes službu cloud	Ano	Ano	Ne	Ano
Kompatibilní zařízení	iOS 8+	Android 4.3+ a iOS 8.2+	Android 4.1+ a iOS 6+	Android 4.4+
Nezávislé aplikace	Ne	Od verze 2.0	Ne	Ne
Ovládání hlasem	Ano	Ano	Ne	Ne
Ovládání gesty zápěstí	Ne	Ano	Ne	Ano

Obr. 2.1: Hodiny Sony Smartwatch 3 [5].

2.3 Bezdrátová komunikace nositelných zařízení

Nejpoužívanějšími technologiemi pro komunikaci nositelných zařízení jsou BLE a Wi-Fi. Většina nositelných zařízení využívá pro komunikaci technologii BLE. Technologie BLE má relativně nízkou přenosovou rychlost max. 1 Mb/s, ovšem energetická náročnost je nízká [1]. Studie zmíněná v článku [27] se zaobírá porovnáním energetické náročnosti technologií BLE a Wi-Fi na mobilním zařízení iPhone. Měření bylo provedeno uvnitř budovy, kdy bylo na mobilním zařízení vypnuto mobilní připojení a všechny aplikace. Výsledek studie uvádí, že použití technologie BLE pro přenos dat prodloužilo výdrž mobilního zařízení až o 30 % oproti Wi-Fi. Některá zařízení, u kterých se očekává větší datový přenos, mohou obsahovat Wi-Fi modul, který je sice energetiky náročnější, ale pokryje požadavky na rychlost a zpoždění přenosu [28].

2.4 Sony Smartwatch 3

Tyto hodinky viz obr. 2.1 využívají platformu Android Wear, která umožňuje kromě notifikací ovládat i funkce chytrého mobilního telefonu. Hodinky lze využívat se zařízením Android 4.3 a novějšími nebo s iOS 8.2 a novějšími. Hodinky vydrží na baterii až dva dny a disponují GPS modulem, vestavěným kompasem, akceleometrem, gyroskopem a stupněm krutí IP68. Lze také změnit jejich vzhled vyjmutím hlavní jednotky a vložením do jiného řemínku [29]. Hardwarové specifikace jsou uvedeny v tab. 2.2.

[5] Zdroj: `goo.gl/Kz9Guk`.

Tab. 2.2: Hardwarové specifikace Sony Smartwatch 3 [29].

Rozměry	$36 \times 51 \times 10$ mm
Váha	38 g
SoC	Qualcomm Snamdragon 400 APQ8026
CPU	ARM Cortex-A7, 1200 MHz, čtyři jádra
GPU	Qualcomm Adreno 305, jedno jádro
RAM	512 MB
Paměť	eMMC 4 GB
Displej	1,6 palce, TFT, 320 × 320 pixelů
Baterie	420 mAh, Li-Polymer, neodjímatelná
Operační systém	Android Wear
USB	Micro USB 2.0
Lokalizace	GPS, A-GPS
Bezdrátové připojení	Wi-Fi 802.11 b/g Bluetooth 4.0 NFC
Senzory	Čidlo snímání okolního světla Akcelerometr Kompas Gyroskop

2.4.1 Párování

Aby bylo možné plně vyžít možností chytrých hodinek, je nutné je nejdříve spárovat s mobilním telefonem. Lze spárovat více chytrých hodinek s jedním telefonem, ale spárovat jedny chytré hodinky s více telefony možné není.

Pro párování je nejprve nutné si stáhnout aplikaci Android Wear app[6], jejíž primární funkcí je konfigurace hodinek a synchronizace aplikací. Dále je nutné zapnout na obou zařízení Bluetooth a povolit viditelnost. V aplikaci se zobrazí seznam dostupných zařízení, po vybrání jednoho z nich se na obou zařízení objeví párovací klíč – je nutné zkontrolovat jejich shodu. Pokud je telefon již spárován s jinými hodinkami, je potřeba v nastavení zvolit možnost spárovat s novým zařízením. Proces párování může trvat i několik minut a jakmile je dokončen zobrazí se zpráva o úspěšném spárování. Je pravděpodobné, že po spárování dojde automaticky k aktualizaci a restartování hodinek [30].

[6] `https://play.google.com/store/apps/details?id=com.google.android.wearable.app`

2.4.2 Synchronizace přes Wi-Fi

Pokud je dokončen proces párování a hodinky disponují Wi-Fi, lze nastavit automatické připojení ke známým Wi-Fi sítím v případě, že dojde ke ztrátě Bluetooth spojení. Dále je nutné v aplikaci Android Wear povolit službu Android Wear cloud sync[7]. Tento proces umožní synchronizaci zařízení pokud jsou připojena k Internetu, nehledě na typ sítě a vzdálenost. Není nutné nastavovat připojení k Wi-Fi síti v hodinkách. Všechny dostupné sítě a jejich přihlašovací údaje jsou automaticky přeneseny do chytrých hodinek a v případě ztráty Bluetooth spojení dojde k připojení k lokální síti. Omezením této služby je nemožnost připojit se k Wi-Fi sítím, které vyžadují návštěvu internetové stránky před přihlášením [31].

[7] `www.support.google.com/androidwear/answer/6213289?hl=en`

3 ANDROID WEAR

3.1 Požadavky

Pro možnost vývoje pro Android Wear platformu je zapotřebí připravit vývojové prostředí dle následujících požadavků:

1. Android studio[1] nejméně verze 0.8.0, lze použít i jiná vývojová prostředí jako Eclipse[2] nebo IntelliJ IDEA[3].
2. Android SDK nejméně verze 4.4W (API level 20), která je první verzí podporující Android Wear.
3. Android support repository verze 4 nebo 13 (verze 13 obsahuje funkce verze 4). Support knihovny poskytují zpětnou kompatibilitu funkcí z novějších verzí, které nejsou v Android frameworku. Tyto knihovny se liší verzí Android, kterou podporují a funkcemi, které zpřístupňují. Verze 4 je určena pro Andoid 2.3 (API level 9) a vyšší, zatímco pro verzi 13 je určena verze Android 2.3 (API level 13) a vyšší. Je třeba použít právě jednu z těchto knihoven, jelikož obsahují třídy pro práci s nositelnými zařízeními příkladem jsou grafické komponenty WearableListView, GridViewPager nebo NotificationManagerCompat pro zaslání notifikací do nositelného zařízení.
4. Android telefon s verzí Android OS nejméně 4.3 (API level 18).

Pro testovací účely je vhodné nainstalovat Android Wear emulátor. Je zapotřebí si na telefonu nainstalovat Android Wear aplikaci, aby bylo možné propojit telefon s emulátorem nebo hodinkami [1].

3.2 Životní cyklus aplikace

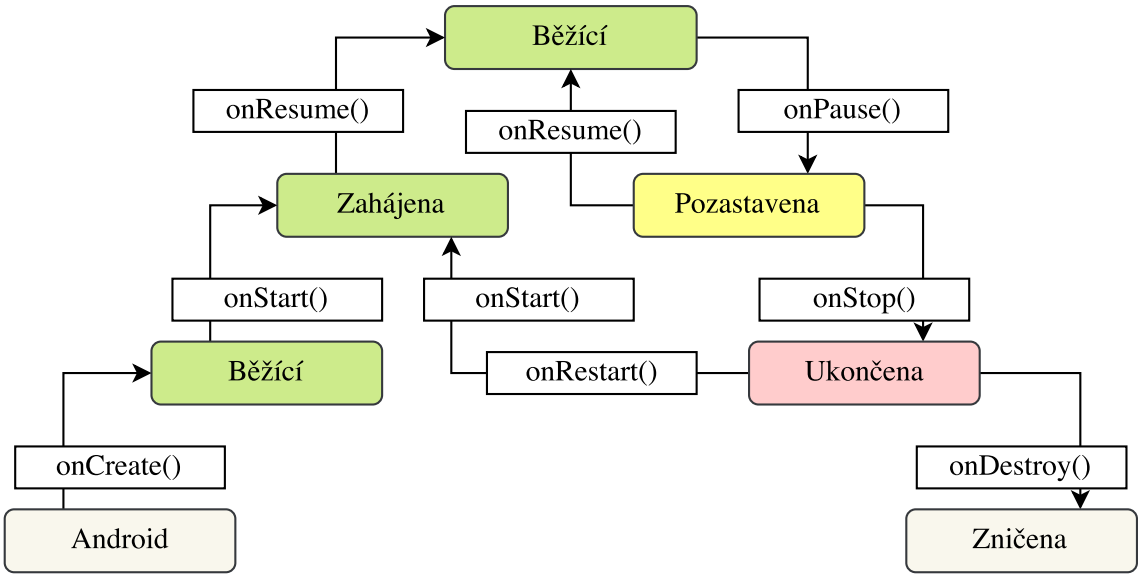
Aplikace s uživatelským rozhraním pro Android Wear fungují stejně jako pro Android. Každá aplikace je tvořena jednou či více instancemi třídy Activity. Třída Activity je komponenta, která uživateli poskytuje obrazovku, se kterou může interagovat. Každá aktivita má vlastní životní cyklus a může po delší dobu existovat ve třech stavech [32].

1. Běží: aktivita je v popředí obrazovky a je aktivní.
2. Pozastavena: aktivita je viditelná, ale je překrytá jinou aktivitou například dialogovými okny. Tato aktivita může být ukončena systémem v případě nedostatku volného místa v paměti.

[1] `www.developer.android.com/studio/index.html`

[2] `www.eclipse.org/`

[3] `www.jetbrains.com/idea/`



Obr. 3.1: Životní cyklus Android aplikace [33].

3. Zastavena: Stav aktivity je držen v paměti, ale není viditelná. Může být ukončena systémem. Aktivita je po spuštění vždy obnovena do výchozího stavu.

Mezi jednotlivými stavy se přechází metodami volanými systémem viz obr. 3.1 [32].

3.3 Synchronizace a výměna dat

Android Wear nabízí pro výměnu dat mezi kapesním a nositelným zařízením tzv. Data Layer API. Toto API se skládá z datových objektů, které systém může posílat a synchronizovat, a posluchačů (Listener), které upozorní aplikace na změnu událostí v datové vrstvě [34].

- **DataItem** – poskytuje datové uložiště, které je automaticky mezi zařízeními synchronizováno.
- **MessageApi** – umožnuje zasílat zprávy. Toto API je vhodné pro jednosměrnou komunikaci typu požadavek nebo komunikaci požadavek/odpověď. Příkladem je RPC (Remote Procedure Call) – volání funkcí na vzdáleném zařízení například ovládání hudebního přehrávače, systémových funkcí nebo spuštění některé aktivity.
- **Asset** – slouží k přenosu velkých binárních dat jako jsou obrázky. Systém bere v potaz šířku pásma Bluetooth, a proto objekty ukládá do mezipaměti, aby nedocházelo k znovu zasílání. Tento proces je zvaný caching.
- **WearableListenerService** – vytvořením objektu, který dědí z této třídy, lze vytvořit službu na pozadí naslouchající událostem Data Layer API. Systém automaticky řídí životní cyklus této služby.

- **DataListener** – také naslouchá událostem Data Layer API, ale je implementován uvnitř aktivity. Vyžitím této třídy lze událostem naslouchat pouze, když uživatel používá aplikaci.
- **ChannelApi** – lze využít pro přenos velkých datových objektů jako hudba a videa. Narozdíl od Assets přenáší data bez automatické synchronizace. Šetří místo v paměťovém uložišti na rozdíl od DataApi, protože před odesláním nevytváří kopii. Umožňuje přenášet streamovaná data například hudba nebo hlasový vstup z mikrofonu.

Pokud má uživatel více nositelných zařízení, dojde k automatické synchronizaci s každým z nich. Systém automaticky synchronizuje data s přímo připojenými zařízeními a cloud službou [34].

3.4 Přehled Android funkcí

V této sekci jsou uvedené některé dostupné programovatelné funkce v Android 4.3 (API level 18) podle [35].

- **Bluetooth** – balíček android.bluetooth obsahuje třídy, které poskytují funkce pro skenování dostupných zařízení v okolí, připojení se k zařízením a ovládání přenosu dat mezi nimi. Toto API podporuje kromě standardního Bluetooth i BLE. Aplikace, která chce komunikovat pomocí Bluetooth skrze tuto API, musí mít BLUETOOTH oprávnění. Některé funkce například jako modifikování nastavení Bluetooth vyžadují BLETOOTH_ADMIN oprávnění.
- **Senzory** – balíček android.hardware umožňuje práci s dostupnými senzory (gyroskop, kompas, ...). V této verzi obsahuje i třídy pro ovládání kamery, které jsou od API level 21 označeny za zastaralé a nahrazeny balíčkem android.hardware.camera2.
- **Média** – balíček android.media obsahuje třídy pro nahrávání zvuku, přehrávání zvuků a videa. Obsahuje také třídu AudioManager, která umožňuje měnit hlasitost zvuku.
- **Wi-Fi** – balíček android.net.Wi-Fi poskytuje informace o dostupných sítích (síla signálu, přenosová rychlost, identifikátor sítě, identifikátor zařízení, IP adresa, MAC adresa, ...). Mezi další funkce patří přidání, uložení, ukončení a zahájení Wi-Fi připojení. Pro zjištění stavu Wi-Fi adaptéru je třeba ACCESS_WIFI_STATE oprávnění a pro změnu CHANGE_WIFI_STATE. Lze také vytvořit P2P (Peer-to-Peer) spojení přes Wi-Fi Direct.
- **NFC** – V balíčku android.nfc jsou třídy pro realizaci komunikace s zařízeními disponujícími technologií NFC.
- **Nastavení** – Pomocí třídy Setting.Secure z balíčku android.provider lze přistoupit k nastavením například (zapnutí lokalizační služby, povolení dat na po-

zadí, ...), které lze měnit s oprávněním WRITE_SECURE_SETTINGS. Bohužel toto oprávnění není zpřístupněno aplikacím třetích stran. Pomocí třídy Settings.System z balíčku android.provider lze získat přístup k nastavením jako je stav Wi-Fi, jas, hlasitost. Pokud má aplikace WRITE_SETTINGS oprávnění, může tato nastavení měnit.

3.5 Možnosti aplikace nezávislé na telefonu

V Android Wear 2.0 mohou aplikace běžet nezávisle na aplikaci v chytrém telefonu. Pokud hodinky nedisponují Wi-Fi modulem, je tok dat přemostěn přes mobilní telefon. Aplikace může přijímat a posílat notifikace skrze internetovou službu FCM (Firebase Cloud Messaging) nebo GCM (Google Cloud Messaging). Lze například využívat protokoly HTTP (Hypertext Transfer Protocol), TCP (Transmission Control Protocol) a UDP (User Datagram Protocol), ale balíček (android.webkit) obsahující funkce pro zobrazování webových stránek není dostupný [33].

Na hodinkách mohou běžet i aplikace, které nepotřebují být stále připojené k mobilnímu telefonu, ale stačí je jednou za čas synchronizovat. Příkladem je aplikace Google Fit[4], která sleduje pohybovou aktivitu. Dále je možné na hodinkách nastavit budík, spustit časovač, zobrazit mapy a spustit navigaci k danému místu s vužitím GPS nebo mobilního připojení.

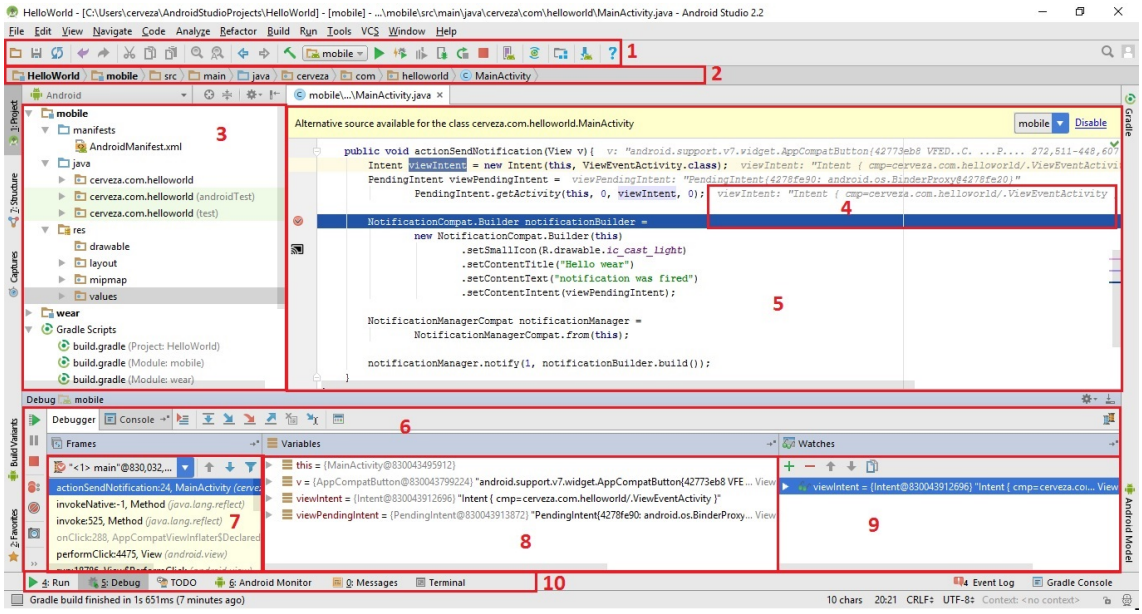
3.6 Android studio

Android Studio je oficiální vývojové prostředí pro vývoj Android aplikací, jehož základ tvoří vývojové prostředí IntelliJ IDEA od firmy JetBrains. IntelliJ IDEA je primárně určené pro vývoj v programovacím jazyce Java, ale podporuje vývoj v jiných jazycích (PHP, Python, ...). Android Studio nabízí kromě standardních funkcí editorů i několik zajímavých funkcí jako Gradle systém pro automatizaci sestavovacího procesu aplikace, emulátor, možnost nahrát změny do běžící aplikace bez vytvoření nového APK souboru, integrovaný GitHub a šablony vzorových kódů [36].

3.6.1 Uživatelské rozhraní

Na obr. 3.2 je vidět vzhled rozhraní prostředí Android Studio, v oblasti kódu je ukázka vyvolání notifikace po kliknutí na tlačítko viz výpis kódu 3.1. Vzhled vyvolané notifikace zobrazené na chytrých hodinkách je vidět na obr. 3.3.

[4]https://play.google.com/store/apps/details?id=com.google.android.apps.fitness



Obr. 3.2: Uživatelské rozhraní android studia.

Popis uživatelského rozhraní

1. Panel nástrojů pro vyvolání akcí projektu. Například: spuštění aplikace, nahrání aplikace do chytrého mobilního telefonu, spuštění debug módu.
2. Navigační záložka zobrazující adresářovou strukturu editovaného souboru. Lze skrze ní přejít do jiného souboru.
3. Okno zobrazující strukturu projektu. Na obr. 3.2 je vidět struktura projektu pro Android Wear aplikaci. Jsou tu dva hlavní adresáře mobile pro aplikaci na telefonu a wear pro aplikaci nositelného zařízení. Každý z adresářů dále obsahuje adresář java obsahující zdrojové kódy aplikace a adresář res který obsahuje obrázky, jazykové lokalizační soubory, xml soubory uživatelského rozhraní aktivit.V kořenovém adresáři je dále adresář manifest s konfigurační souborem AndroidManifest.xml. Manifest soubor musí obsahovat každá aplikace, protože obsahuje informace, které potřebuje znát systém před spuštěním aplikace. V tomto souboru jsou popsány komponenty aplikace (aktivity, služby...), deklarovány požadovaná přístupová práva k chráněným systémovým funkcím.
4. V debug módu lze vidět na řádku s proměnou její stav.
5. Okno pro zobrazení zdrojových kódů.
6. Zde lze zobrazit dodatečná okna například pro ladící mód.
7. Přehled volaných metod aplikace v okně ladícího módu.
8. Souhrn všech proměnných aktivity.
9. Okno zobrazující sledované proměnné a výrazy.
10. Lišta pro přepínaní dodatečných oken například ladící mód, terminál.

Výpis 3.1: Ukázka kódu zaslání notifikace.

```
Intent intent = new Intent(this, ViewEventActivity.class);
PendingIntent viewPendingIntent =
        PendingIntent.getActivity(this, 0, intent, 0);

NotificationCompat.Builder notificationBuilder =
        new NotificationCompat.Builder(this)
                .setSmallIcon(R.drawable.ic_cast_light)
                .setContentTitle("Test")
                .setContentText("testovaci upozorneni")
                .setContentIntent(viewPendingIntent);

NotificationManagerCompat notificationManager =
        NotificationManagerCompat.from(this);

notificationManager.
        notify(1, notificationBuilder.build());
```

Obr. 3.3: Záznam obrazovky vyvolané notifikace na hodinkách.

4 VYTVOŘENÁ APLIKACE

Úkolem praktické části předkládané bakalářské práce je navrhnout a vytvořit aplikaci pro ovládání systémových funkcí chytrého mobilního telefonu pomocí chytrých hodinek. Pro vypracování byly zvoleny chytré hodinky Sony Smartwatch 3, jelikož disponují technologií Wi-Fi pro bezdrátovou komunikaci a využívají platformu Android Wear.

Z důvodu složitější kompozice grafického rozhraní, byly vytvořeny dvě aplikace. První nazvaná Toggles, která primárně umožňuje přepínání systémových funkcí a zobrazení základních informací o stavu Wi-Fi připojení, baterie a zobrazení mapy s aktuální polohou. Druhá aplikace nazvaná Bluetooth manager, která zobrazuje aktuální nastavení adaptéru, spárovaná zařízení, připojená zařízení a zařízení v dosahu.

V této části je popsány obě vytvořené aplikace. Toggles, její funkce, vzhled a struktura projektu. Bluetooth manager, její funkce, vzhled a struktura projektu.

4.1 Aplikace Toggles

4.1.1 Funkce

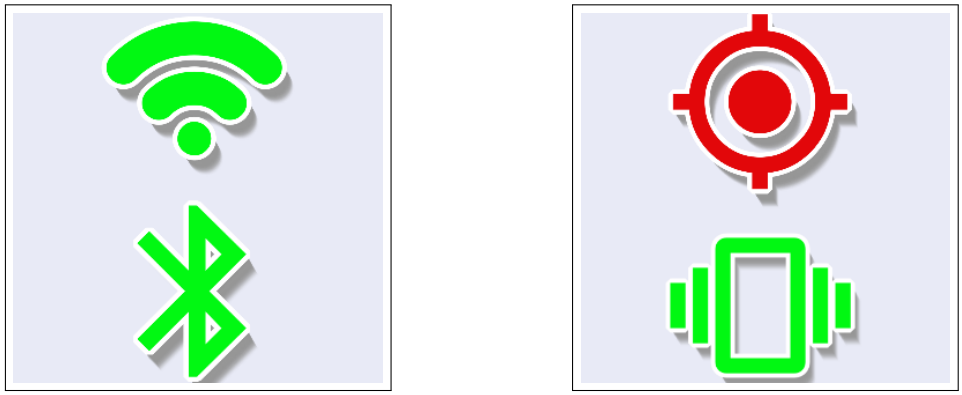
- Možnost zapnout/vypnout Wi-Fi, mobilní data, GPS, Bluetooth.
- Přepínání tří stavů zvuku (vypnuto, zapnuto, vibrace).
- Zobrazení dodatečné informace k Wi-Fi připojení (síla signálu, identifikátor sítě, IP a MAC adresa).
- Zobrazení stavu baterie (kapacita, zdraví, status – nabíjení/vybíjení).
- Automatická synchronizace stavu funkcí v telefonu s hodinkami.
- Zobrazení mapy s aktuální polohou a výpisem aktuálních souřadnic (zeměpisná délka, zeměpisná šířka).

4.1.2 Vzhled

Aplikace obsahuje několik pohledů (fragmentů), které jsou přepínány pomocí výsuvného menu.

- **Navigační menu** – skládá se ze čtyř záložek Baterry, Toggles, Wifi info a Map. Viz obr. 4.7
- **Mobilní funkce** – v tomto pohledu je vidět list tlačítek sloužící k přepínání stavu funkcí v telefonu. Tlačítka mění svůj vzhled na základě aktuálního stavu funkce v telefonu. Viz obr. 4.1.

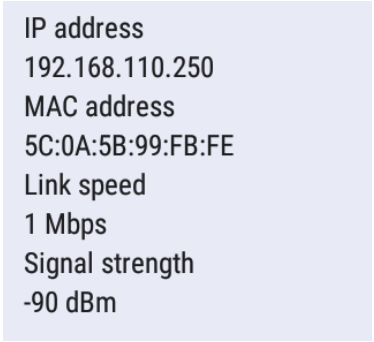
- **Informace o Wi-Fi připojení** – v tomto pohledu jsou veškeré informace o Wi-Fi připojení, které lze ze systému získat (SSID, BSSID, IP adresa, MAC adresa, šířka pásma připojení, síla signálu)[1] viz obr. 4.2.
- **Mapa** – v této aktivitě je zobrazena mapa s textovým polem, kde je uvedena zeměpisná poloha v GPS souřadnicích. Stisknutím tlačítka na levé straně obrazovky je mapa přesměrována na aktuální pozici. V horní části aplikace se nachází ovládání pro přiblížení a oddálení mapy viz obr. 4.3. Odejít z této aktivity je možné pomocí dlouhého stisknutí obrazovky (až tři vteřiny) a následného potvrzení dialogu viz obr. 4.5.
- **Potvrzovací dialog** – v případě pokusu o vypnutí Wi-Fi nebo Bluetooth je zobrazen dialog, který uživatele upozorňuje, že by mohlo dojít ke ztrátě spojení s mobilním telefonem. Uživatel má možnost do pěti vteřin akci potvrdit. Pokud tak neučiní, akce se neprovede a dialog se zavře viz obr. 4.4.
- **Informační dialog** – tento dialog slouží k upozornění uživatele, že je nutné akci dokončit v telefonu. Ovládání systémových funkcí telefonu (GPS, mobilní data[2], ...) lze pouze pokud uživatel vlastní root oprávnění. V opačném případě lze zobrazit pouze systémový dialog, kde je možné danou funkci ovládat. Tento dialog slouží k upozornění uživatele, že je nutné dokončit akci v chytrém zařízení viz obr. 4.6.
- **Baterie** – v tomto pohledu je zobrazen aktuální stav baterie telefonu. Zobrazené informace jsou kapacita, zdraví, status (připojení ke zdroji napájení), zdroj napájení (USB, bezdrátové, ze sítě).



Obr. 4.1: Vzhled úvodní aktivity v hodinkách.

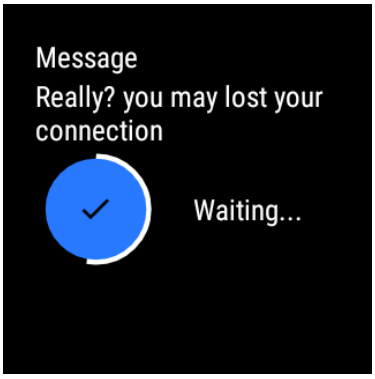
[1] Zobrazit frekvenci a přenosový kanál je možno až od verze Android 5.0.

[2] Do verze Android 4.* je možné ovládat tuto funkci bez root oprávnění využitím chyby v systému `goo.gl/DlxHg8`.



Obr. 4.2: Vzhled zobrazení Wi-Fi informací.

Obr. 4.3: Vzhled zobrazení mapy.



Obr. 4.4: Vzhled potvrzovacího dialogu.

4.1.3 Struktura projektu

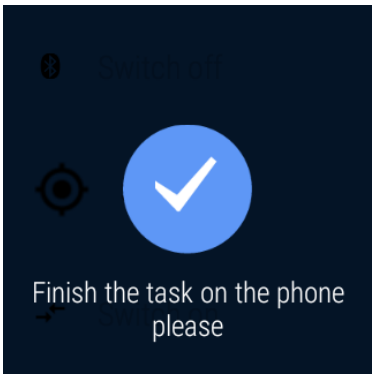
Projekt se skládá ze tří modulů MobileApp, WearApp a mwAPI. Modul MobileApp obsahuje aplikaci mobilního telefonu zatímco WearApp části aplikace pro chytré hodinky. Modul mwAPI obsahuje třídy a konstanty společné pro oba zmíněné moduly.

MobileApp

V tomto modulu jsou obsaženy veškeré potřebné třídy pro běh části aplikace v mobilním telefonu.

- **ListenerService** – je služba, která dědí od WearableListenerService. Tato služba je zaregistrována systémem, který ji předává zprávy přijaté od nositelných zařízení. Služba dále předává zprávy službě nazvané ToggleFunctionServer.
- **ToggleFunctionServer** – tato služba pracuje na principu serveru, kdy zpracovává požadavky z hodinek, provede odpovídající akci a zašle odpověď. Pokud tato služba neběží, je automaticky spuštěna službou ListenerService při obdržení zprávy. Při spuštění dochází k registraci dvou receiverů FunctionChangeReceiver a BatteryStatusReceiver, které naslouchají změně systémo-

Obr. 4.5: Dialog opuštění mapy.



Obr. 4.6: Dialog jdi do telefonu.

Obr. 4.7: Část navigačního menu.

vých funkcí a při změně některé z nich vyvolají synchronizaci s chytrými hodinkami. K ukončení služby a odregistrování receiverů dojde po jedné minutě od doručení poslední zprávy.

- **FunctionChangeReceiver** – naslouchá systémovým zprávám o změně stavu Wi-Fi, Bluetooth, GPS, nastavení zvuku a mobilního připojení.
- **BatteryStatusReceiver** – naslouchá změně stavu baterie. Po přijetí notifikace o změně, vytvoří objekt MyBatteryInfo a zašle zprávu hodinkám pomocí služby MessageSendService.
- **MessageSendService** – zajišťuje odesílání zpráv hodinkám v odlišném vlákně.
- **FunctionChangeHandler** – obsahuje funkce pro přepínání systémových funkcí a zaslání synchronizačních údajů viz synchronizace stavu funkcí.

WearApp

V tomto modulu jsou obsaženy veškeré potřebné třídy potřebné pro běh části aplikace na chytrých hodinkách.

- **MainActivity** – definuje hlavní vzhled aplikace (navigační menu a kontejner pro zobrazování jednotlivých pohledů). Umožňuje přepínaní jednotlivých

pohledů (fragmentů). Přijímá zprávy zaslané z aplikace v telefonu, které dále předává zodpovědným fragmentům.

- **MapsActivity** – zajišťuje připojení k Google map service, zobrazení mapy a aktualizaci polohy zařízení. Zobrazuje aktuální polohu na mapě a výpis GPS souřadnic.
- **NavigationAdapter** – definuje zobrazení ikon v navigačním menu (vzhled ikony, text a akci, která bude vyvolána po vybrání daného prvku v menu).
- **IOnNavigationChange** – je rozhraní, které musí být implementováno v hlavní aktivitě, aby mohla být upozorněna na vybraní jiného prvku v navigaci.
- **FragmentBattery** – pohled, který zobrazuje informace o baterii v telefonu. Zpracovává zprávy o změně stavu baterie a aktualizuje pohled.
- **FragmentToggles** – pohled, který zobrazuje seznam tlačítek, pro přepínaní funkcí v telefonu. Při přijetí zprávy o změně funkcí, aktualizuje vzhled tlačítek. Odesílá odpovídající zprávy při kliknutí na jednotlivá tlačítka.
- **FragmentWifiInfo** – pohled, který zobrazuje informace o Wi-Fi připojení. Každý dvě vteřiny vyžaduje aktualizaci, zasláním zprávy.
- **MessageSendServiceWear** – umožňuje zasílání zpráv aplikaci v telefonu v odděleném vlákně.

mwAPI

V tomto modulu jsou obsaženy veškeré třídy používané jak v chytrém telefonu tak i na chytrých hodinkách.

- **SharedConstants** – obsahuje konstanty definující cestu jednotlivých zpráv. Pomocí těchto konstant dochází na obou stranách k identifikaci zprávy a předání odpovědné komponentě. Dále obsahuje list používaných přepínačů funkcí.
- **MyBatteryInfo, MyWifiInfo** – datové objekty pro zasílaní informací mezi zařízeními.
- **ControlFunction** – třída reprezentující jednotlivé přepínače funkcí v FragmentToggles.
- **EMessageType** – výčtový typ definující bližší určení typu zprávy (změna akce, synchronizace, ...).
- **SerializeUtils** – třída, která zajišťuje konverzi objektů například MyWifiInfo do pole bytů a zpět. Toto je nutné provést, aby bylo možné přenést data pomocí messageAPI do druhého zařízení.

4.1.4 Běh aplikace

Algoritmus 1: Běh části aplikace přepínání funkcí na hodinkách.

```
1  navázání připojení;
2  nastavení posluchače zpráv;
3  synchronizace stavu funkcí telefonu;
4  nastavení grafické rozhraní;

5  if stisk tlačítka then
6  |  zašli požadavek;
7  end
8  if odpověď == synchronizuj then
9  |  změň nastavení tlačítka;
10 end
```

Algoritmus 2: Běh aplikace v mobilním telefonu.

```
1  if nová zpráva then
2  |  zjisti typ zprávy;
3  |  vykonej odpovídající akci;
4  end
5  if Změna stavu funkce v telefonu then
6  |  Zjištění aktuálních stavu funkcí;
7  |  Zaslání informací o aktuálním stavu;
8  end
```

4.1.5 Synchronizace stavu funkcí

Pro synchronizaci stavu funkcí byl zvolen způsob, kdy na straně telefonu běží služba, která naslouchá systémovým notifikacím, o změně daných funkcí pomocí BroadcastReciever. Kdykoliv obdrží zprávu o změně stavu některé funkce, například Wi-Fi upozorní službu FunctionChangeHandlerService, aby provedla synchronizaci viz výpis kódu A.1 řádky 1–9 . Tato služba si pamatuje stav funkcí o kterých ví aplikace v hodinkách, zjistí si nový stav funkcí a do aplikace v hodinkách zašle jen informace o změněných funkcí. Funkce zjišťující změněné stavy funkcí viz výpis kódu A.1 řádky 10–17. Tento způsob šetří kapacitu přenosového pásma, protože při změně jen jedné funkce zašle méně dat, v současnosti je to jedna čtvrtina množství co by bylo odesláno, kdyby se vždy posílala informace o všech funkcích. Toto řešení způsobilo

v případě přechodu hlavní aktivity aplikace v hodinkách do stavu pozastavena špatnou synchronizaci, protože aplikace nezpracovala informaci o změně některé funkce a v tu chvíli pamatovaný stav funkcí v telefonu se lišil s uloženým stavem funkcí v hodinkách. To se může stát při zobrazení mapy, informací o Wi-Fi nebo zobrazení některého dialogu. Pro eliminaci této vady byla provedena změna, aby si hlavní aktivita v hodinkách vyžádala vždy při přechodu do aktivního stavu (funkce cyklu onResume()) synchronizaci viz výpis kódu A.1 řádky 18–28. V rámci této synchronizace dojde na obou stranách k nastavení stavu funkcí do výchozího stavu.

Synchronizaci by šlo řešit i automatickým dotazováním aplikací v hodinkách o synchronizaci. V tomto případě by interval dotazování měl být nastaven do jedné sekundy, aby uživatel neměl pocit, že došlo k chybě, protože se nic neděje a nepokusil se vyvolat danou akci znovu. Tento způsob řešení by vedl k větším nárokům na přenosové pásmo a energetickému vytížení hodinek.

Další řešení synchronizace, které bylo vzato v potaz, je v případě vyvolání požadované akce v telefonu zaslat zprávu o její změně. Bohužel u akcí, na které nemá uživatel oprávnění a musí je potvrdit v telefon, nelze v aplikaci zjistit zda akci potvrdil nebo zrušil. Zjištění výsledku lze pouze pomocí BroadcastReceiver.

4.1.6 Typy zpráv

Ve výčtovém typu EMessageType je seznam všech používaných zpráv. Každá zpráva se skládá z prefixu a specifického názvu. V projektu jsou použité dva typy prefixů definované ve třídě SharedConstants. Důvodem přidání prefixů bylo zredukování množství podmínek v hlavní aktivitě a použití pouze jedné funkce pro zpracování zpráv na rozhraní aktivity a fragmentu. Fragment s tlačítky pro přepínání stavu funkcí v telefonu používá celkem sedm zpráv. V tomto případě hlavní aktivita zkontroluje zda daná zpráva začíná daným prefixem, poté zavolá funkci handleMessage na fragmentu a ten na základě celé cesty zprávy rozhodne co udělat. Druhý prefix mají společný fragmenty pro zobrazení informací o Wi-Fi a baterii, přestože nejsou v jednom fragmentu. Důvodem je fakt, že došlo k jejich rozdělení, a jelikož každý fragment zpracovává pouze jednu zprávu, bylo to takto ponecháno.

4.2 Aplikace Bluetooth manager

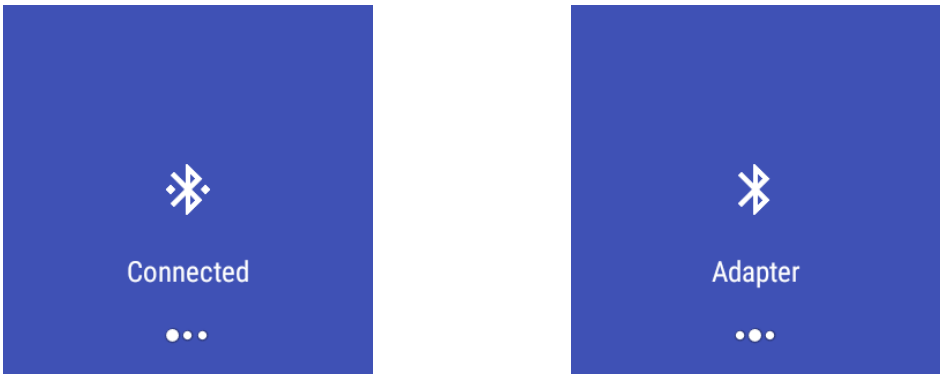
4.2.1 Funkce

- Zobrazení nastavení adaptéru (název, MAC adresa).
- Zobrazení identifikátorů spárovaných zařízení.
- Zobrazení a detekce identifikátorů připojených zařízení.
- Zobrazení a detekce identifikátorů zařízení v dosahu.

4.2.2 Vzhled

Aplikace je rozdělena do tří pohledů (fragmentů), mezi kterými je možné přepínat pomocí výsuvného menu.

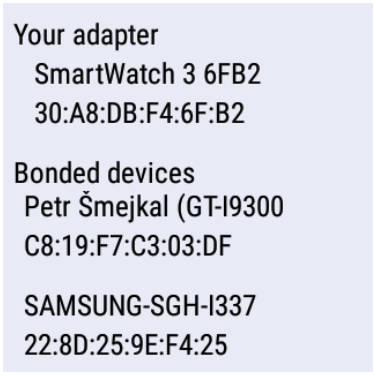
- **Navigační menu** – skládá se ze tří záložek connected, adapter, discovered. Viz obr. 4.8.
- **Úvodní obrazovka** – zobrazuje identifikátory adaptéru a seznam všech spárovaných zařízení. Viz obr. 4.9.
- **Připojená zařízení** – zobrazuje seznam připojených zařízení. Pokud dojde k připojení nebo odpojení zařízení, projeví se změna aktualizací seznamu. Viz obr. 4.10.
- **Nalezená zařízení** – zobrazuje seznam všech zařízení v okolí. Pokud přibude některé zařízení v dosahu, projeví se změna aktualizací seznamu. Viz obr. 4.11.



Obr. 4.8: Část navigačního menu.

4.2.3 Struktura projektu

Aplikace interaguje s Bluetooth adaptérem v hodinkách a nepotřebuje komunikovat s telefonem. To je důvodem proč v tomto projektu je pouze jeden modul app, který obsahuje všechny potřebné třídy pro běh aplikace v chytrých hodinkách.



Obr. 4.9: Nastavení adaptéru

Obr. 4.10: Připojená zařízení

Obr. 4.11: Nalezená zařízení

- **AdapterBluetoothList** – adaptér, který přizpůsobuje informace o Bluetooth zařízení pro zobrazení v seznamu.
- **MainActivity** – obsahuje navigační menu a kontejner pro pohledy (fragmenty). Řídí přepínání pohledů v kontejneru na základě vybrané položky v menu.
- **Adaptéry** – zobrazují a získávají data pro pohledy. Aplikace obsahuje celkem tři pohledy BluetoothAdapterFragment, BluetoothConnectedFragment a BluetoothDiscoveredFragment. Každý z pohledů obsahuje logiku pro interakci s adaptérem za účelem získání potřebných dat.

4.3 Navigační menu

V této části je popsán způsob implementace navigačního menu za pomoci WearableDrawerLayout a WearableNavigationDrawer. Obě třídy jsou součástí Android knihoven.

Drawer layout je hlavní kontejner, který umožňuje vysunutí definovaných elementů (např. navigačního menu) z horní nebo spodní strany aplikačního okna.

Navigation drawer umožňuje uživateli přepínat různé pohledy aplikace. Menu je automaticky přístupné v horní části okna a k jeho vysunutí dojde v případě, kdy uživatel dosáhne na začátek rolovacího obsahu.

4.3.1 Implementace

Implementace navigačního menu se skládá ze tří hlavních kroků. Vytvoření grafického rozhraní obsahující navigační menu, přizpůsobení potřebných dat (obrázek, popisek) pomocí adaptéru a jejich propojení v hlavní aplikaci.

Grafické rozhraní

V grafickém rozložení hlavní aplikace (activity_main.xml) je nutné definovat výše uvedené dvě třídy. Drawer layout musí být kořenovým prvkem, který obsahuje jak navigační menu tak i samostatný obsah stránky viz Příloha A.2. Pokud je obsah stránky umístěn v prvku, který podporuje rolování, musí tento prvek také implementovat rozhraní NestedScrollingChild, které zajistí upozornění nadřazeného prvku drawer layout na událost rolování a vysunutí navigačního menu.

Adaptér

Adaptér musí dědit od třídy WearableNavigationDrawerAdapter viz Příloha A.3, jelikož tato třída obsahuje čtyři hlavní funkce, které je potřeba přepsat, aby došlo ke správnému zobrazení prvků menu.

- **getCount()** – vrací celkový počet prvků, které menu obsahuje.
- **onItemSelected(int position)** – je zavolána pokud uživatel vybere v menu jinou položku.
- **getItemText(int pos)** – je zavolána vždy při vykreslení každé položky a měla by vracet odpovídající popisek.
- **getCount(int pos)** – je také zavolána při vykreslení každé položky a měla by vracet ikonu, která se zobrazuje nad obrázkem.

Všechny ikony začínají prefixem „ic_nav_“ a zbytek názvu tvoří jejich popisek. Tento způsob umožňuje dynamické načítání obrázků. V pžíloze A.3 na řádku 30 a 31 je vidět vytvoření názvu z prefixu a popisku. Dále na řádcích 32 až 34 ověření existence a získání identifikátoru obrázku pomocí systémových funkcí. A v poslední řadě načtení a vrácení požadovaného obrázku na řádku 35.

V případě, že uživatel vybere v menu jinou položku, je nutné upozornit hlavní aplikaci na změnu položky a vyvolání odpovídající akce např. přepnutí pohledu. Proto je potřeba vytvořit vlastní rozhraní (v tomto případě IOnNavigatinChange s jednou funkcí onNavigationChange(int position)) a implementovat ho v hlavní

aplikaci. Aby adaptér mohl zavolat danou funkci, je nutné mu v konstruktoru předat objekt implementující dané rozhraní (v tomto případě hlavní aplikace) viz Příloha A.3 řádky kódu 8 až 14. Po tomto kroku je již možné upozornit aplikaci na změnu a předat ji pozici vybrané položky viz řádky kódu 20 až 23.

Propojení adaptéru s navigačním menu

Propojení adaptéru a navigačního menu je možné vidět v ukázce kódu viz Příloha A.4, která obsahuje zkrácenou část kódu hlavní aplikace. Propojení se zkládá ze tří kroků. Za prvé získání instance NavigationDrawer z grafického rozhraní viz řádky 12 až 14. Za druhé vytvoření instance adaptéru viz řádky 15 a 16. Za třetí nastavení adaptéru v instanci NavigationDrawer viz řádky 17 a 18.

Na řádku 21 je funkce z rozhraní IOnNavigationChange, která je vyvolána adaptérem při změně položky v menu.

4.4 Fragmenty a přepínání pohledů

V této části je popsáno využití fragmentů v aplikaci a jejich využití při přepínaní pohledů. Využití fragmentů má několik výhod: možnost dekompozice složitější aplikace na menší celky, jednoduší výměna a sdílení dat mezi fragmenty než mezi aktivitami, možnost vrátit předchozí stav aplikace tlačítkem zpět při využití zásobníku.Při práci s fragmenty jsou důležité následující třídy.

- **Fragment** – je část uživatelského rozhraní, které musí být umístěno v aktivitě. Každý fragment má vlastní životní cyklus podobný cyklu aktivity. Přestože má vlastní cyklus, je závislý na cyklu aktivity. Pokud dojde k uspání aktivity, dojde k uspání i všech fragmentů, které obsahovala.
- **FragmentManager** – je rozhraní pro práci s fragmenty (přepínání fragmentů, práci se zásobníkem fragmentů, . . .).
- **FragmentTransaction** – je třída jejíž instanci lze získat prostřednictvím FragmentManager. Transakce reprezentuje konečný počet operací s fragmenty. Obsahuje funkce pro výměnu, schování, odebrání, přidání fragmentů. V jedné transakci je například možné vyměnit dva fragmenty, třetí schovat a čtvrtý úplně odebrat. Po provedení této transakce je nutné opět získat novou transakci pro další operace s fragmenty.

4.4.1 Životní cyklus fragmentů

Každý fragment prochází od vytvoření po ukončení celkem dvanácti funkcemi.

1. **onAttach** – je volána při přiřazení fragmentu aktivitě.
2. **onCreate** – je volána za účelem prvotní inicializace fragmentu.

3. **onCreateView** – je volaná za účelem vytvoření pohledu fragmentu a jeho přidání do hierarchie grafického rozhraní.
4. **onActivityCreated** – je volaná ve chvíli, kdy přiřazená aktivita dokončila volání funkce onCreated().
5. **onViewStateRestored** – je volaná ve chvíli, kdy je obnoven stav jeho hierarchie pohledu.
6. **onStart, onResume, onPause, onStop** – jsou ekvivalentní stejnojmenným funkcím cyklu aktivity.
7. **onDestroyView** – umožňuje fragmentu uvolnit zdroje alokované pro vytvoření pohledu.
8. **onDestroy** – je ekvivalentí stejnojmenné funkci aktivity.
9. **onDetach** – je volaná ve chvíli, kdy dojde k odebrání fragmentu z aktivity.

4.4.2 Implementace

Při práci s fragmenty je vždy nutné provést následující čtyři kroky: definovat grafické rozhraní, kontejner obsahující fragment, vytvořit vlastní třídu fragmentu a implementovat přepínání fragmentů.

Grafické rozhraní

V grafickém rozhraní fragmentu je možné použít stejné prvky jako v rozhraní aktivity s tím rozdílem, že toto rozhraní bude mít stejnou velikost jako jeho kontejner.

Aplikace Bluetooth Manager obsahuje fragment nazvaný BluetoothAdapterFragment jehož rozhraní se skládá ze čtyřech textových polí a jednoho RecycleView viz Příloha A.7.

Kontejner

V grafickém rozložení aktivity je možné specifikovat fragment použitím značky „<fragment>“ nebo některým z typů rozložení (LinearLayout, RelativeLayout, . . .). V prvním případě je nutné specifikovat atribut „android:name“ s názvem třídy vytvořeného fragmentu a systém automaticky vytvoří instanci každého fragmentu. V druhém případě je třeba specifikovat atribut „android:id“, v kódu vytvořit instanci fragmentu a přiřadit ho prvku se zvoleným identifikátorem.

V obou vytvořených aplikací byl zvolen druhý způsob, kdy grafické rozhraní aktivity obsahuje RelativeLayout s atributem id rovný „@+id/fragment_container“. Viz řádky kódu 7 až 12 v Příloze A.2.

Vlastní třída Fragment

Každý z fragmentů v obou aplikacích dědí od třídy Fragment a obsahuje pouze několik funkcí cyklu (onAttach, onCreate, onResume, onCreateView). Fragmenty obsahují dvě funkce onAttach, které se liší argumentem viz řádky kódu 7 až 15 v Příloze A.6. Je to z důvodu, že funkce s argumentem Activity je v systémech Android pouze do API level 23 a funkce s argumentem Context až od API level 23.

Ve funkci onCreateView je nutné vložit odpovídající rozhraní fragmentu do rozhraní celé aplikace viz řádky 30 a 31. Dále je v této funkci možné získat instance objektů z rozhraní za účelem ovládání obsahu v kódu. V tomto případě byly do lokální promněných uloženy dva objekty TextView viz řádky 32 až 35.

Přepínání pohledů

Aby bylo možné přepínat pohledy fragmentů, je nutné nejdříve získat instanci třídy FragmentManager a instanci všech použitých fragmentů viz řádky 14 až 19 v Příloze A.5. Ve chvíli, kdy je zavolána funkce pro přepnutí fragmentu (v tomto případě funkce onNavigationChange, která je volaná adaptérem navigačního menu) je nutné provést následující kroky.

- Získat instanci transakce z FragmentManager viz řádky 23 a 24.
- Na základě přijatého parametru rozhodnout, který fragment bude vložen viz řádky 26 až 33.
- Přidat transakci operaci, která má být s daným fragmentem provedena (v tomto případě záměna fragmentu v kontejneru za nový) viz řádky 34 a 35.
- Posledním krokem je potvrzení transakce viz řádek 36.

4.5 Recycle view

V případě dynamické práce s větším množství datových sad, kdy jen část dat je zobrazena nebo práce s daty, které se často mění, je výhodné použít recycle view. Recycle view poskytuje efektivní implementaci pro práci s daty tohoto typu.

V této části je popsána implementace grafického rozhraní fragmentu pro přepínání funkcí telefonu. Pro implementaci jsou nejdůležitější čtyři třídy RecycleView, LayoutManager, RecyclerView.Adapter a RecyclerView.ViewHolder. RecycleView je kontejner pro zobrazení definovaných položek. LayoutManager se stará o zobrazení definovaných položek (lineárně pod sebou, ve dvou sloupcích, ...) a rozhoduje, kdy je potřeba načíst nové položky. Pokud je potřeba načíst nové položky, manager volá adaptér, který má za úkol vytvořit požadovaný počet položek na základě datové sady. Každá položka je reprezentována třídou ViewHolder. V této třídě je možné definovat vlastní vzhled a chování pro každou položku.

4.5.1 Položky – ViewHolder

V obou aplikací je vždy ViewHolder definován ve třídě Adaptér, aby mohl přistupovat k jeho proměnným. ViewHolder je kontejner pohledu jednotlivé položky a proto musí být definován v odděleném xml souboru. V tomto případě list_item.xml viz ukázka kódu A.9. Každá položka obsahuje pouze jeden obrázek s identifikátorem circle viz řádky 11 až 16.

Aby bylo možné v kódu pracovat s obrázkem z pohledu, je nutné ho ve třídě ItemViewHolder zpřístupnit vytvořením odpovídající proměnné viz ukázka kódu A.12 řádek 5 a v konstruktoru ho načíst pomocí identifikátoru viz řádky 8 a 9. Aby bylo možné vyvolat akci při kliknutí na obrázek, musí ItemViewholder implementovat View.OnClickListener a v konstruktoru nastavit sebe jako odposlouchávač události kliknutí viz řádek 10. Ve funkci onClick je vidět implementace odezvy na událost kliknutí. Na řádcích 15 až 17 je získán typ položky (EMessageType), na řádcích 19 až 28 je zobrazení dialogu o možné ztrátě připojení v případě, kdy se uživatel pokouší vypnout Bluetooth nebo Wi-Fi. V poslední fázi na řádcích 30 až 35 je odeslání zprávy do chytrého telefonu o přepnutí funkce, která odpovídá dané položce.

4.5.2 Adaptér

Datovou sadu reprezentuje list objektů třídy ControlFunction, který je předán konstruktorem. V adaptéru je nutné přepsat tři rodičovské funkce. První je onCreateViewHolder, která vrací instanci třídy ItemViewHolder viz ukázka kódu A.10 řádky 21 až 26. V této funkci je také načten požadovaný vzhled položky z xml souboru, který je předán ItemViewHolder konstruktorem viz řádky 24 a 25. Druhá funkce je getItemCount, která vrací celkový počet položek z datové sady viz řádky 28 až 31. Poslední funkcí je onBindViewHolder, která naplní požadovanou instanci ItemViewHolder daty viz ukázka kódu A.11.

Hlavním účelem pohledu fragmentu s ikonami pro přepínání je měnit vzhled několika ikon podle stavu funkce stavu v připojeném mobilním telefonu. Tato funkce je implementována ve výše zmíněné funkci onBindViewHolder. Název každé ikony se skládá z prefixu ic_icno_ názvu ikony (audio, gps ...) a sufixu podle stavu funkce (_on, _off, _vibration[3]). V ukázce kódu A.11 na řádcích 8 až 10 je vidět získání názvu ikony z datové sady a nahrazení případných mezer v názvu podtržítkem. Na řádcích 12 až 22 je přidán sufix odpovídající stavu dané funkce. Na řádku 23 je přidán prefix. Na řádcích 24 až 26 je získán identifikátor ikony podle názvu. A poslední krok

[3]Tohoto stavu může nabývat pouze ikona pro ovládání zvuku.

na řádku 28 je odpovídající ikona nastavena jako zdroj obrázku v pohledu pro danou položku.

4.5.3 Propojení s fragmentem

Aby došlo k zobrazení obsahu RecycleView, je potřeba ho definovat v definici pohledu hlavní aplikace nebo fragmentu viz ukázka kódu A.8. V ukázce kódu A.13 je možné vidět poslední kroky implementace. Je nutné získat instanci RecycleView z pohledu viz řádky 13 a 14. Vytvořit instanci adaptéru a předat mu datovou sadu viz řádek 17. A v poslední řadě nastavit instanci RecycleView adaptér viz řádek 19 a layout manager viz řádky 20 a 21.

5 ZÁVĚR

V teoretické části bakalářské práce byl popsán operační systém Android a Android Wear z hlediska fungování a možností pro ovládání funkcí chytrého mobilního telefonu, získaní základních informací o nastavení, vývoj aplikace ve vývojovém prostředí Android Studio a možnosti a implementace komunikace dvou zařízení. Na základě získaných informací byla vyvinuta Androird Wear aplikace pro ovládání funkcí spárovaného telefonu.

Praktická část bakalářské práce byla zaměřena na implementaci ovládání základních funkcionalit, pochopení fungování systému Android (Android Wear) z hlediska výměny dat mezi zařízeními, interakce a práci s daty mezi komponentami (hlavní aplikace, fragmenty) aplikace.

Z důvodů rostoucího počtu funkcí došlo k rozdělení aplikace na dvě samostatné aplikace. První nazvaná Toggles, jelikož primární funkcí této aplikace je přepínání funkcí v chytrém mobilním telefonu. A druhá aplikace nazvaná Bluetooth manager, jelikož pracuje s Bluetooth adaptérem a zobrazuje informace získané z adaptéru.

Aplikace Toggles umožňuje ovládat WiFi, Bluetooth, zvuk (zapnuto, vypnuto, vibrace), GPS a mobilní data. V případě přepínání GPS je nutné přejít do mobilního telefonu a potvrdit dialog, protože GPS je systémová funkce a nelze ji ovládat bez root oprávnění. K přepnutí mobilních dat doje pouze ve verzi Android KitKat (na tuto verzi byl primárně zaměřen vývoj) díky vyžití chyby v systému. Tato chyba byla odstraněna v Android Lollipop, a proto ji není možné v novějších verzí ovládat bez root oprávnění. Dále je možné zobrazit informace o aktuálně připojené Wi-Fi síti, zjistit informace o stavu baterie a zobrazit na mapě aktuální polohu. Aplikace automaticky synchronizuje stav funkcí v telefonu, pokud je aplikace v hodinkách zapnuta nebo v telefonu dojde ke změně některé z funkcí.

Aplikace Bluetooth Manager pracuje pouze s Bluetooth adaptérem v chytrých hodinkách, a tudíž není zapotřebí, aby aplikace komunikovala s mobilním zařízení. V této aplikaci zjistit název adaptéru včetně MAC adresy a seznamu všech spárovaných zařízení. Další funkcí této aplikace je skenování okolí a zobrazení všech Bluetooth zařízení v dosahu. Poslední funkcí je možnost zobrazení seznamu připojených zařízení. Bohužel poslední funkce nezobrazuje zařízení, která byla připojena již před spuštěním této funkcionality. Důvodem je fakt, že operační systém Android neposkytuje seznam připojených zařízení, ale pouze naslouchat události připojení nebo odpojení Bluetooth zařízení.

Vytvořením aplikací Toggles a Bluetooth manager bylo dosaženo hlavních cílů bakalářské práce.

LITERATURA

[1] RUIZ, David Cuartielles. *Professional android wearables.* Indianapolis, IN: John Wiley and Sons, 2014. ISBN 1118986857.

[2] ČÍHAL, Josef. *Simulace H2H a M2M komunikace v mobilní síti LTE-Advanced.* Vysoké učení technické v Brně, 2015.

[3] LIU, Charles Z. a Manolya KAVAKLI. *Data-Aware QoE-QoS Management.* Macquaire University Sydney, NSW 2109, Australia, 2016.

[4] BOSWARTHICK, David, Omar ELLOUMI a Olivier HERSENT. *M2M communications: a systems approach.* Hoboken, N.J.: Wiley, 2012. ISBN 978-1-119-99475-6.

[5] ASADI, Arash, QING WANG a Vincenzo MANCUSO. *A Survey on Device-to-Device Communication in Cellular Networks. IEEE Communications Surveys & Tutorials* [online]. 2014, 16(4), 1801-1819 [cit. 2016-10-28]. DOI: 10.1109/COMST.2014.2319555. ISSN 1553877x. Dostupné z: <goo.gl/1NMjHj>.

[6] OMETOV, Aleksandr, Pavel MASEK, Jani URAMA, Jiri HOSEK, Sergey ANDREEV a Yevgeni KOUCHERYAVY.*Implementing secure network-assisted D2D framework in live 3GPP LTE deployment.* In: 2016 IEEE International Conference on Communications Workshops (ICC) [online]. Tampere University of Technology, Finland, Tampere, Korkeakoulunkatu 10, Brno University of Technology, Czech Republic, Brno, Technicka 3082/12: IEEE, 2016, s. 749-754 [cit. 2016-10-28]. DOI: 10.1109/ICCW.2016.7503877. ISBN 9781509004485. Dostupné z: <http://ieeexplore.ieee.org/document/7503877/>.

[7] *Timeline of Telecommunications. Telephone tribute* [online]. [cit. 2016-10-28]. Dostupné z: <http://www.telephonetribute.com/timeline.html>

[8] LUETH, Knud Lasse.*Why the Internet of Things is called Internet of Things: Definition, history, disambiguation* [online]. In: . 2014 [cit. 2016-10-28]. Dostupné z: <goo.gl/VC8Erh>

[9] *Juniper Networks, Inc.*, 2014 MACHINE-TO-MACHINE(M2M)-THE RISE OF THE MACHINES , 2011.

[10] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015–2020 White Paper. In:*Cisco*[online]. [cit. 2016-10-28]. Dostupné z: <www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html>

[11] MILLER, Brent A. a Chatschik. BISDIKIAN. *Bluetooth revealed*. 2nd ed. Upper Saddle River, NJ: Prentice Hall PTR, c2002. ISBN 0130672378.

[12] CLARK, Bryan. *The Differences Between Bluetooth 4.0 and Wi-Fi Direct You Need To Know. In: Makeuseof* [online]. [cit. 2016-10-28]. Dostupné z: <goo.gl/IEmNvp>

[13] OMETOV, Aleksandr, Pavel MASEK, Lukas MALINA, et al. Feasibility characterization of cryptographic primitives for constrained (wearable) IoT devices. *In: 2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)* [online]. IEEE: IEEE, 2016, s. 1-6 [cit. 2016-10-28]. DOI: 10.1109/PERCOMW.2016.7457161. ISBN 9781509019410. Dostupné z: <http://ieeexplore.ieee.org/document/7457161/>

[14] A. ALDERSON. *Sports Tech: Fitness Trackers. Engineering & Technology* [online]. 2015, 10(4), 84-5 [cit. 2016-10-28]. DOI: 10.1049/et.2015.0461. ISSN 17509637. Dostupné z: <http://digital-library.theiet.org/content/journals/10.1049/et.2015.0461>

[15] BODHANI, A. *The connected body. Engineering & Technology* [online]. 2015, 10(4), 44-7 [cit. 2016-10-28]. DOI: 10.1049/et.2015.0417. ISSN 17509637. Dostupné z: <http://digital-library.theiet.org/content/journals/10.1049/et.2015.0417>

[16] *What's New in watchOS 3* [online]. In: . [cit. 2017-04-24]. Dostupné z: <www.macrumors.com/roundup/watchos-3/>

[17] KAMPL, Bernd. *The 5 most important smartwatch platforms for developers* [online]. In: . [cit. 2016-10-29]. Dostupné z: <goo.gl/UTCcf4>

[18] CATANZARITI, Patrick. *Smartwatch Platforms to Consider Developing for in 2015*[online]. In: . [cit. 2016-10-29]. Dostupné z: <https://www.sitepoint.com/smartwatch-platforms-to-consider-developing-for-in-2015/>

[19] Tizen [online]. [cit. 2016-10-29]. Dostupné z: <https://www.tizen.org/about>

[20] *Asteroidos* [online]. [cit. 2016-12-13]. Dostupné z: <https://asteroidos.org/>

[21] KESSLER, Derek. LG's new Urbane LTE smartwatch runs webOS. In: *Webosnation* [online]. [cit. 2016-12-13]. Dostupné z:<http://www.webosnation.com/lgs-new-urbane-lte-smartwatch-runs-webos>

[22] 4 top smartwatch platforms. In: *Gadgetsnow* [online]. [cit. 2016-12-13]. Dostupné z: <http://www.gadgetsnow.com/4-top-smartwatch-platforms-watchos-android-tizen-rtos/checklistshow/54364038.cms>

[23] WatchOS Human Interface Guidelines. *apple* [online]. [cit. 2016-11-12]. Dostupné z:<https://developer.apple.com/watchos/human-interface-guidelines>

[24] Android Wear: The essential guide. In: *Wareable* [online]. [cit. 2016-11-12]. Dostupné z: <http://www.wareable.com/android-wear/what-is-android-wear-comprehensive-guide>

[25] Introduction to Pebble Original and Steel. *Getpebble* [online]. [cit. 2016-11-12]. Dostupné z: <https://help.getpebble.com/customer/portal/articles/1722567-introduction-to-pebble>

[26] Tizen Developers. *Tizen* [online]. [cit. 2016-11-12]. Dostupné z: <https://developer.tizen.org/development/api-guides/native-application/uix/voice-control/voice-control>

[27] PUTRA, Guntur Dharma, Azkario Rizky PRATAMA, Alexander LAZOVIK a Marco AIELLO. Comparison of energy consumption in Wi-Fi and bluetooth communication in a Smart Building. In: *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)* [online]. Groningen: IEEE, 2017, s. 1-6 [cit. 2017-05-14]. DOI: 10.1109/CCWC.2017.7868425. ISBN 9781509042289. Dostupné z: <http://ieeexplore.ieee.org/document/7868425/>

[28] OMETOV, Aleksandr, Ekaterina OLSHANNIKOVA, Pavel MASEK, Thomas OLSSON, Jiri HOSEK, Sergey ANDREEV a Yevgeni KOUCHERYAVY. *Dynamic Trust Associations over Socially-Aware D2D Technology: A Practical Implementation Perspective. IEEE Access* [online]. 2016, , 1-1 [cit. 2016-10-28]. DOI: 10.1109/ACCESS.2016.2617207. ISSN 21693536. Dostupné z: <http://ieeexplore.ieee.org/document/7604102/>

[29] Sony Smartwatch 3. *Sonymobile* [online]. [cit. 2016-11-10]. Dostupné z: <http://www.sonymobile.com/cz/products/smart-products/smartwatch-3-swr50/>

[30] Pair your watch with your phone. *Google* [online]. [cit. 2016-11-10]. Dostupné z: <https://support.google.com/androidwear/answer/6056630?hl=en&ref_topic=6056389>

[31] Connect to Wi-Fi. *Google* [online]. [cit. 2016-11-10]. Dostupné z: <https://support.google.com/androidwear/answer/6207505?hl=en>

[32] Android app lifecycle. *Android developer portal* [online]. [cit. 2016-10-29]. Dostupné z: <https://developer.android.com/training/basics/activity-lifecycle/starting.html>

[33] Standalone app android wear. *Android developer portal* [online]. [cit. 2016-10-29]. Dostupné z: <https://developer.android.com/wear/preview/features/standalone-apps.html>

[34] Sending and Syncing Data. *Android* [online]. [cit. 2016-11-11]. Dostupné z: <https://developer.android.com/training/wearables/data-layer/index.html>

[35] Package Index. *Android* [online]. [cit. 2016-11-11]. Dostupné z: <https://developer.android.com/reference/packages.html>

[36] Meet Android Studio. *Android developer portal*. [cit. 2016-10-29]. Dostupné z: <https://developer.android.com/studio/intro/index.html#project_structure>

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

API	Application Programming Interface
BLE	Bluetooth Low Energy
BTS	Base Transciever Station
D2D	Device-to-Device
H2H	Human-to-Human
HTTP	HyperText Trasfer Protocol
IoE	Internet of Everything
IoT	Internet of Things
IrDA	Infrared Data Association
LTE	Long Term Evolution
M2M	Machine-to-Machine
NFC	Near Field Communication
ProSec	Proximity Service
QoE	Quality of Experience
QoS	Quality of Service
RFID	Radio Frequency Identification
RPC	Remote Procedure Call
SDK	Software Development Kit
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

SEZNAM PŘÍLOH

A UKÁZKA ZDROJOVÉHO KÓDU

Výpis A.1: Ukázka kódu synchronizace stavu funkcí.

```
1  public void onReceive(Context context, Intent intent) {
2    Intent intentSendMessage =
3    new Intent(context, FunctionChangeHandlerService.class)
4      .setAction("WEARABLE_COMMAND")
5      .putExtra("MESSAGE_TYPE",
6      EMessageType.FUNCTIONS_STATE_SYNCHRONIZATION);
7    context.startService(intentSendMessage);
8  }
9  public List<ControlFunction> getUpdatedControls() {
10   List<ControlFunction> newControls =
11      checkFunctionStates();
12   List<ControlFunction> differentControls =
13   new ArrayList<>(newControls);
14   differentControls.removeAll(actualControls);
15   actualControls = newControls;
16   return differentControls;
17 }
18 protected void onResume() {
19   super.onResume();
20   Intent intent;
21   intent = new Intent(this,MessageSendServiceWear.class);
22     .setAction("SEND_MESSAGE")
23     .putExtra("MESSAGE_PATH",MESSAGE_PATH)
24     .putExtra("BYTES",
25     serialize(EMessageType.FUNCTIONS_STATE_INITZIALIZE));
26   this.startService(intent);
27 }
```

Výpis A.2: Definice hlavního layoutu apliakce

```
<android.support.wearable.view.drawer.WearableDrawerLayout
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- Zacatek hlavniho obsahu okna -->
    <RelativeLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:id="@+id/fragment_container">
    </RelativeLayout>
    <!-- Konec hlavniho obsahu okna -->

    <android.support.wearable.view.drawer
      .WearableNavigationDrawer
        android:id="@+id/top_navigation_drawer"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</android.support.wearable.view
        .drawer.WearableDrawerLayout>
```

Výpis A.3: Adaptér navigačního menu

```
1  public class NavigationAdapter extends
2  WearableNavigationDrawer.WearableNavigationDrawerAdapter{
3
4      private Context mContext;
5      private IOnNavigationChange
6                  onNavigationChangeListener;
7      private ArrayList<String> items;
8
9      public NavigationAdapter(Context context) {
10         mContext = context;
11         if(context instanceof IOnNavigationChange){
12             onNavigationChangeListener =
13                   (IOnNavigationChange) context;
14         }
15     }
16
17     public int getCount() {
18         return items.size();
19     }
20
21     public void onItemSelected(int position) {
22         onNavigationChangeListener.
23             onNavigationChange(position);
24     }
25
26     public String getItemText(int pos) {
27         return items.get(pos);
28     }
29
30     public Drawable getItemDrawable(int pos) {
31         String imageName = "ic_nav_".
32             concat(items.get(pos).toLowerCase());
33         int resID = mContext.getResources().
34             getIdentifier(imageName , "drawable",
35             mContext.getPackageName());
36         return mContext.getDrawable(resID);
37     }
38 }
```

Výpis A.4: Propojení adaptéru s uživatelským rozhraním

```
1  public class MainActivity extends Activity
2    implements IOnNavigationChange{
3
4      private WearableNavigationDrawer
5        mWearableNavigationDrawer;
6
7      @Override
8      protected void onCreate(Bundle savedInstanceState) {
9          super.onCreate(savedInstanceState);
10         setContentView(R.layout.activity_main);
11
12         mWearableNavigationDrawer =
13     (WearableNavigationDrawer)
14     findViewById(R.id.top_navigation_drawer);
15         NavigationAdapter navigationAdapter =
16       new NavigationAdapter(this);
17         mWearableNavigationDrawer.
18       setAdapter(navigationAdapter);
19     }
20
21     public void onNavigationChange(int pos) {
22       //on navigation change
23     }
24 }
```

Výpis A.5: Přepínání pohledů

```
1  public class MainActivity extends Activity
2   implements IOnNavigationChange{
3
4      BluetoothAdapterFragment bluetoothAdapterFragment;
5      BluetoothDiscoveredFragment
6              bluetoothDiscoveredFragment;
7
8      FragmentManager fragmentManager;
9      FragmentTransaction fragmentTransaction;
10
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_main);
14
15         fragmentManager = getFragmentManager();
16
17         bluetoothAdapterFragment =
18             new BluetoothAdapterFragment();
19         bluetoothDiscoveredFragment =
20             new BluetoothDiscoveredFragment();
21     }
22
23     public void onNavigationChange(int pos) {
24         fragmentTransaction =
25             fragmentManager.beginTransaction();
26         Fragment fragment = bluetoothAdapterFragment;
27         switch (pos){
28             case 0:
29                 fragment = bluetoothConnectedFragment;
30                 break;
31             case 1:
32                 fragment = bluetoothAdapterFragment;
33                 break;
34         }
35         fragmentTransaction.
36             replace(R.id.fragment_container,fragment);
37         fragmentTransaction.commit();
38     }
39 }
```

Výpis A.6: Vlastní třída dědící od třídy Fragment

```
1  public class BluetoothAdapterFragment extends Fragment{
2      Context context;
3
4      TextView textName;
5      TextView textAddress;
6
7      public void onAttach(Context context) {
8          this.context = context;
9          super.onAttach(context);
10     }
11
12     public void onAttach(Activity activity) {
13         context = activity.getApplicationContext();
14         super.onAttach(activity);
15     }
16
17     public void onCreate(Bundle savedInstanceState) {
18         //TODO on create
19         super.onCreate(savedInstanceState);
20     }
21
22     public void onResume() {
23         super.onResume();
24         //TODO on resume
25     }
26
27     public View onCreateView(LayoutInflater inflater,
28         ViewGroup container, Bundle savedInstanceState) {
29
30         View view = inflater.inflate(R.layout.
31             bluetooth_adapter_fragment, container, false);
32         textName = (TextView)view.
33             findViewById(R.id.adapter_name);
34         textAddress = (TextView)view.
35             findViewById(R.id.adapter_address);
36         return  view;
37     }
38 }
```

Výpis A.7: Grafické rozhraní fragmentu BluetoothAdapterFragment

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="5dp">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Your␣adapter"
        android:textColor="@color/black"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/adapter_name"
        android:textColor="@color/black"
        android:layout_marginLeft="10dp"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/adapter_address"
        android:textColor="@color/black"
        android:layout_marginLeft="10dp"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="10dp"
        android:text="Bonded␣devices"
        android:textColor="@color/black"/>

     <android.support.wearable.view.WearableRecyclerView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/bonded_devices_list">
     </android.support.wearable.view.WearableRecyclerView>
</LinearLayout>
```

Výpis A.8: Grafické rozložení fragmentu toggles

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android=
    "http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">

    <android.support.wearable.view.WearableRecyclerView
        android:id="@+id/toggle_list"
        android:orientation="vertical"
        android:layout_gravity="center_horizontal"
        android:layout_height="match_parent"
        android:layout_width="wrap_content">
    </android.support.wearable.view.WearableRecyclerView>

</LinearLayout>
```

Výpis A.9: Grafické rozložení jedné položky recycle view

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android=
        "http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_centerHorizontal="true">

    <ImageView android:id="@+id/circle"
        android:layout_height="100dp"
        android:layout_width="100dp"
        android:layout_marginLeft="40dp"
        android:adjustViewBounds="false"
        android:cropToPadding="false" />
</LinearLayout>
```

Výpis A.10: Adaptér recycle view část 1

```
1  public class Adapter extends RecyclerView.Adapter {
2      private List<ControlFunction> mDataset;
3      private final Context mContext;
4      private final LayoutInflater mInflater;
5
6      public Adapter(Context context,
7              List<ControlFunction> dataset) {
8          mContext = context;
9          mInflater = LayoutInflater.from(context);
10         mDataset = dataset;
11
12     }
13
14     public void refresh(List<ControlFunction> mDataset)
15     {
16         this.mDataset = mDataset;
17         notifyDataSetChanged();
18     }
19
20     @Override
21     public RecyclerView.ViewHolder
22     onCreateViewHolder(ViewGroup parent, int viewType) {
23         return new ItemViewHolder(mInflater.inflate
24             (R.layout.list_item, null));
25     }
26
27     @Override
28     public int getItemCount() {
29         return mDataset.size();
30     }
```

Výpis A.11: Adaptér recycle view část 2

```
    @Override
    public void onBindViewHolder(RecyclerView.ViewHolder
        holder, int position) {
        ItemViewHolder itemHolder =
            (ItemViewHolder) holder;

        String imageName =
            mDataset.get(position).getName()
            .toLowerCase().replace("␣","_");

        switch (mDataset.get(position).getState()){
            case 0:
              imageName = imageName.concat("_off");
              break;
            case 1:
              imageName = imageName.concat("_on");
              break;
            case 2:
              imageName = imageName.concat("_vibration");
              break;
        }
        imageName = "ic_icon_".concat(imageName);
        int resID = mContext.getResources()
            .getIdentifier(imageName , "drawable",
             mContext.getPackageName());

        itemHolder.imageView.setImageResource(resID);
        holder.itemView.setTag(position);
    }
}
```

Výpis A.12: Item view holder

```
1  private class ItemViewHolder
2  extends RecyclerView.ViewHolder
3  implements View.OnClickListener
4  {
5      private ImageView imageView;
6      public ItemViewHolder(View itemView) {
7          super(itemView);
8          imageView = (ImageView)
9          itemView.findViewById(R.id.circle);
10         itemView.setOnClickListener(this);
11     }
12
13     @Override
14     public void onClick(View v) {
15         Integer tag = (Integer) v.getTag();
16         EMessageType messageType =
17             mDataset.get(tag).getID();
18
19         if(messageType ==
20                EMessageType.BLUETOOTH_CHANGE_STATUS ||
21             messageType ==
22                EMessageType.WIFI_CHANGE_STATUS){
23             if(mDataset.get(tag).getState() == 1){
24                 alertDialog("Really? you may lost your"
25                 "connection", messageType.getPath());
26                 return;
27             }
28         }
29
30         Intent intent = new Intent(mContext,
31             MessageSendServiceWear.class)
32                 .setAction("SEND_MESSAGE")
33                 .putExtra("MESSAGE_PATH",
34                  messageType.getPath(0);
35         mContext.startService(intent);
36     }
37 }
```

Výpis A.13: Fragment pracující s recycle view

```
1  public class FragmentToggles extends Fragment {
2
3      private List<ControlFunction> listElements;
4      private Adapter adapter;
5
6      @Nullable
7      @Override
8      public View onCreateView(LayoutInflater inflater,
9          ViewGroup container, Bundle savedInstanceState) {
10         View view = inflater.inflate(R.layout.
11             fragment_toggles, container, false);
12
13         RecyclerView recyclerView = (RecyclerView)
14             view.findViewById(R.id.toggle_list);
15         recyclerView.setNestedScrollingEnabled(true);
16
17         adapter =
18             new Adapter(getActivity(), listElements);
19
20         recyclerView.setAdapter(adapter);
21         recyclerView.setLayoutManager(
22             new LinearLayoutManager(getActivity()));
23
24         return view;
25     }
26 }
```

B OBSAH PŘILOŽENÉHO CD

Přiložené médium obsahuje bakalářskou práci v elektronické podobě a zdrojové kódy vytvořených aplikací. Aplikace byly vytvořeny ve vývojovém prostředí Android Studio 2.2.2.